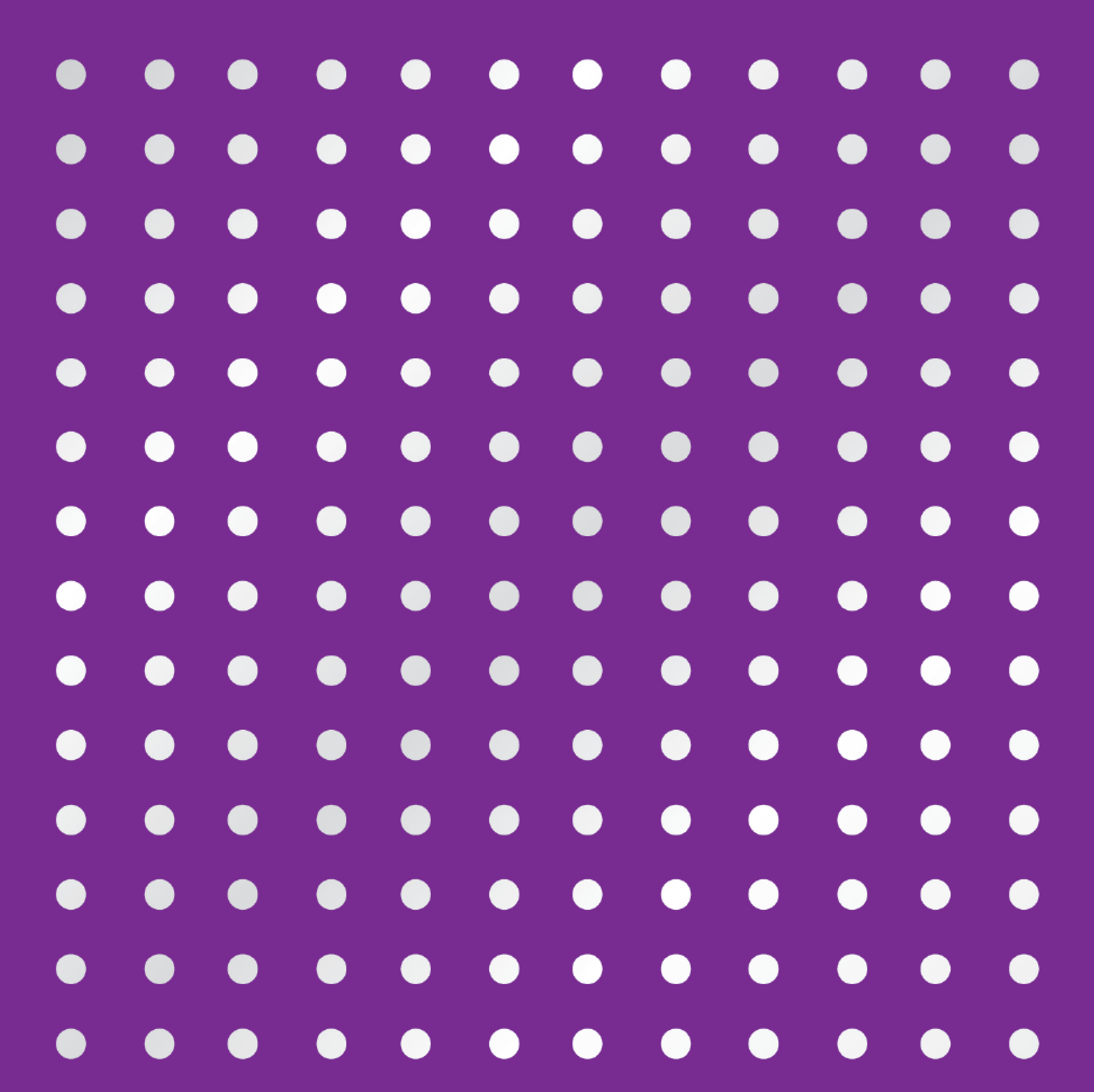
高等院校信息技术规划教材

可视化开发Android应用程序

——拼图开发模式App Inventor

王向辉 张国印 谢晓芹 编著



清华大学出版社

高等院校信息技术规划教材

可视化开发Android应用程序
——拼图开发模式App Inventor

王向辉　张国印　谢晓芹　编著

清华大学出版社

北京

内 容 简 介

本书介绍了一种崭新的 Android 开发模式，将繁琐的代码开发变为轻松的拼图游戏，不仅可以简化开发过程、降低开发难度，还可以提高开发效率，让开发者在 Android 应用程序开发过程中充满乐趣。

本书分为 8 章，涉及的内容包括 App Inventor 的开发环境搭建、用户界面开发、数据存储与访问、动画与游戏、地图应用开发等方面，较全面地覆盖了 Android 程序开发所涉及的内容。

本书内容丰富，实用性强，既可用作高等院校信息技术的教材，也可供相关专业人士参考。

图书在版编目(CIP)数据

可视化开发 Android 应用程序：拼图开发模式 App Inventor /王向辉等编著．--北京：清华大学出版社，2013

高等院校信息技术规划教材

ISBN 978-7-302-32888-9

Ⅰ. ①可…　Ⅱ. ①王…　Ⅲ. ①移动终端—应用程序—程序设计—高等学校—教材　Ⅳ. ①TN929.53

中国版本图书馆 CIP 数据核字(2013)第 136396 号

责任编辑：袁勤勇　薛　阳
封面设计：傅瑞学
责任校对：时翠兰
责任印制：李红英

出版发行：清华大学出版社
　网　　址：http://www.tup.com.cn，http://www.wqbook.com
　地　　址：北京清华大学学研大厦 A 座　　**邮　　编**：100084
　社 总 机：010-62770175　　**邮　　购**：010-62786544
　投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
　质 量 反 馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
印 刷 者：北京富博印刷有限公司
装 订 者：北京市密云县京文制本装订厂
经　　销：全国新华书店
开　　本：185mm×260mm　　**印　　张**：15.25　　**字　　数**：351 千字
版　　次：2013 年 9 月第 1 版　　**印　　次**：2013 年 9 月第 1 次印刷
印　　数：1～2000
定　　价：29.00 元

产品编号：052618-01

前言

Foreword

Android是当今应用最为广泛的智能手机平台，具有丰富的软件资源。Android软件开发具有一定的难度，一般需要开发者具备一定的软件开发知识和经验，App Inventor的出现，将非程序人员编写Android应用软件的愿望变成了现实。App Inventor创造的拼图开发方式，简化了复杂的程序编码过程，极大地提升了学习者对软件编程的兴趣，并为初学者创造了一个轻松的开始。

书中所涉及的内容包括App Inventor的开发环境搭建、用户界面开发、数据存储与访问、动画与游戏、地图应用开发等方面，较全面地覆盖了Android程序开发所涉及的内容。

全书共分8章：

第1章介绍App Inventor的起源、优势与不足，展示出App Inventor开发的一些作品，并对互联网上的App Inventor学习资源进行简单的介绍。

第2章详细说明App Inventor开发环境的安装和账号注册方法，并简单地介绍如何使用模拟器和手机进行调试。

第3章介绍开发App Inventor应用程序的基础知识和基本方法，说明App Inventor的界面编辑器和模块编辑器的作用和使用方法。

第4章详细介绍App Inventor程序开发的基础内容，包括条件判断、循环、列表和函数，并简单说明如何区分模块的事件、属性和方法。

第5章介绍App Inventor界面设计和开发的方法，重点介绍常见的基础控件、媒体控件和社交控件，并对屏幕的布局方式进行讲解。

第6章介绍使用App Inventor开发游戏与动画的方法，详细讲解画布、精灵和球体的控件使用，并介绍碰撞检测的原理。

第7章介绍App Inventor数据存储机制，主要讲解TinyDB控件的存储、读取、更新和删除数据的方法。

第8章介绍App Inventor进行地图应用开发的方法，讲解如何

使用位置传感器和谷歌地图的方法，并简单介绍通知控件。

本书由哈尔滨工程大学王向辉、张国印、谢晓芹负责主要编写工作，参与本书编写和校核工作的还有孙宇彤、杨月、宁凡强、张鑫彧、何志昌和李晓光，这里对他们的辛苦工作表示衷心的感谢。

同时感谢谷歌（中国）的朱爱民先生、东北大学的李丹程和刘莹老师，感谢他们对Android教学和科研工作的帮助，以及对哈尔滨工程大学Android人才培养基地的支持。

App Inventor是一个新兴的开发模式，很多方面还在不断地完善和变化。由于能力和水平所限，虽然竭尽全力，但书中仍然难免存在不足和疏漏的地方，希望各位专家、教师和同学能毫不保留地提出所发现的问题，与编者共同讨论，编者的邮箱为wangxianghui@live.cn。

App Inventor屏蔽了Android程序开发中复杂的编程细节，因此可供没有程序基础的低年级学生和非计算机专业学生学习，可以在大学一年级和二年级开设这门课程。

所有示例代码和教学资源（教学大纲、教学PPT、习题答案等）均可在清华大学出版社（http://www.tup.tsinghua.edu.cn）或哈尔滨工程大学的网站（http://android.hrbeu.ed.cn）上下载。

编　者

2013年4月于哈尔滨工程大学

目录 Contents

第 1 章 chapter 1

Android 与 App Inventor

App Inventor 是一个基于网络、可拖曳的 Android 程序开发环境,它将枯燥的代码变成一块一块的拼图,使 Android 软件开发变得简单有趣,使不懂编程的用户也可以开发出属于自己的 Android 应用程序。通过本章的学习,读者将了解 Android 系统的特点和发展趋势,掌握 App Inventor 的起源、优势和不足,了解互联网上的一些 App Inventor 学习资源。

本章学习目标:

- 了解 Android 系统的起源和发展趋势;
- 掌握 App Inventor 的优势和不足;
- 了解 App Inventor 学习资源。

1.1 Android 简介

Android 是 Google 发布的基于 Linux 平台的开源手机操作系统。Android 一词的本义是“机器人”,国内多称为“安卓”或“安致”。Android 最初应用在智能手机和平板电脑上,是第一个完整、开放、免费的手机操作系统。Android 界面如图 1.1 所示。



图 1.1 Android 界面

Android由安迪·罗宾(Andy Rubin)创建于2003年,于2005年被Google收购。2007年11月5日,Google公司正式向外界展示了这款名为Android的操作系统。

Android是当今世界上应用最广泛的智能手机平台。截止到2012年10月,Android的应用软件已有约70万个,在Google Play的软件下载量达到了250亿次。2012年9月的统计数据表明,每天的Android设备激活数量已经超过了130万台,总激活的设备量超过4.8亿。Android在全球智能手机操作系统市场所占的份额为76%,在中国市场的占有率高达90%。

Android应用得如此广泛,与其自身的特点是分不开的。首先,Android是一款开源的手机操作系统,有效地缩短了开发周期,降低了开发成本。用户可以免费下载Android源代码,并在原系统的基础上进行二次开发,创造具有个性化的Android系统。Android为各种应用程序提供平等的性能支持,能够满足用户对不同应用的需求,用户可以下载自己喜欢的软件到手机上,也可更改手机的界面或图片的浏览方式,使自己的手机与众不同。借助谷歌在互联网运营方面的优势,谷歌地图、邮件和搜索等服务直接内置在Android系统中,作为用户与互联网之间的重要纽带,增强了手机的实用功能。

Android系统最初只是为智能手机所设计,但随着应用领域的不断拓展,Android系统被广泛应用在平板电脑、电视、手表、眼镜、冰箱、耳机和跑步机等设备中,使人们的生活变得越来越智能化。



图1.2 谷歌眼镜

"谷歌眼镜"是谷歌公司在2012年4月发布的一款"扩展现实"眼镜产品,如图1.2所示,可以语音拍照、视频通话和辨明方向,也可以访问互联网信息、处理文字信息和电子邮件。眼镜的右眼镜片上安装了一个微型投影仪和一个摄像头,投影仪用以显示数据,摄像头用来拍摄视频和照片,再通过传感器进行存储和传输,而操控模式可以是语言或触控。

基于Android系统的i'm Watch智能手表,可以显示时间和天气预报,还可以显示短信息和联系人等,并可与其他Android系统手机连接,如图1.3所示

联想智能电视尺寸达到了55英寸,运行最新的Android 4.0 Ice Cream Sandwich系统,内置500万像素摄像头,支持SD卡插槽,如图1.4所示。



图1.3 i'm Watch智能手表



图1.4 联想K91智能电视

三星 Android 冰箱是一台有着内置应用软件的冰箱，功能包括显示照片、播放音乐和给家人留便条等，三星 Android 冰箱还有一个专门用来除霜以及改变温度的应用软件，如图 1.5 所示。

Admiral Touch 耳机同样是基于 Android 系统，配备了一块 2.4 英寸的彩色触摸屏，用户可以用它玩游戏和看电影，这款耳机支持 2.4G 和蓝牙通信，具备 7.1 虚拟环绕音效，如图 1.6 所示。



图 1.5　三星 Android 冰箱



图 1.6　Admiral Touch 耳机

1.2　App Inventor 起源

Android App Inventor 曾是谷歌实验室的一个子计划，于 2010 年 7 月推出，是一款所见即所得的 Android 应用创建器，它允许没有编程知识的用户，通过拖曳特定的应用程序行为模块来创建 Android 应用。2011 年 8 月，谷歌公司将该项目的源代码对外开放，并于 2012 年 1 月将该项目移交给麻省理工学院移动学习中心(The MIT Center for Mobile Learning)，由麻省理工学院的 Hal Abelson 教授领导开发，并与 2012 年 3 月向互联网用户开放使用，更名为 MIT App Inventor，如图 1.7 所示。

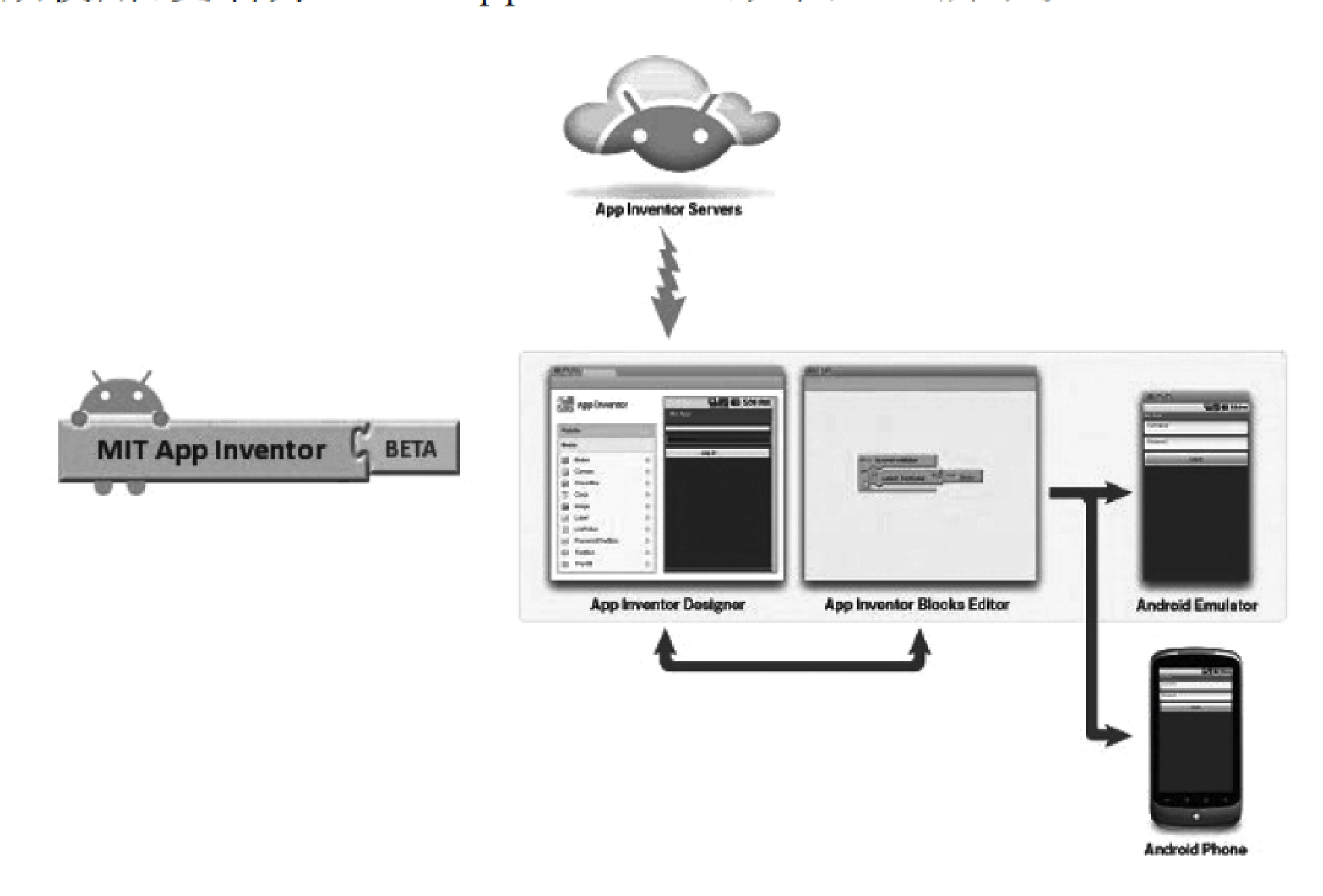


图 1.7　MIT App Inventor

App Inventor 的研发目标是“使人们在移动通信的世界里成为创造者，而不仅是消费者”。App Inventor 创造的拼图开发方式，简化了复杂的程序编码过程，极大地提升了学习者对软件编程的兴趣，并为初学者创造了一个轻松的开始。

App Inventor 的模块编辑语言是基于 Open Blocks 项目的研究成果创建的，而 Open Blocks 项目与 StarLogo TNG、Scratch 这两个项目联系紧密。对 App Inventor 模块编辑器的界面构想，在很多方面借鉴了 StarLogo TNG 和 Scratch 的研究成果，下面分别对这两个项目进行介绍。

StarLogo TNG(The Next Generation)是一种基于主体的仿真语言，由麻省理工媒体实验室和教师教学计划共同研发，其设计主要目的是针对计算机教育，可以用来模拟分散式控制系统的行为，如图 1.8 所示。StarLogo TNG 能够利用开放式图形库提供 3D 视野，并运用模块图形语言来增强易用性和易学性。

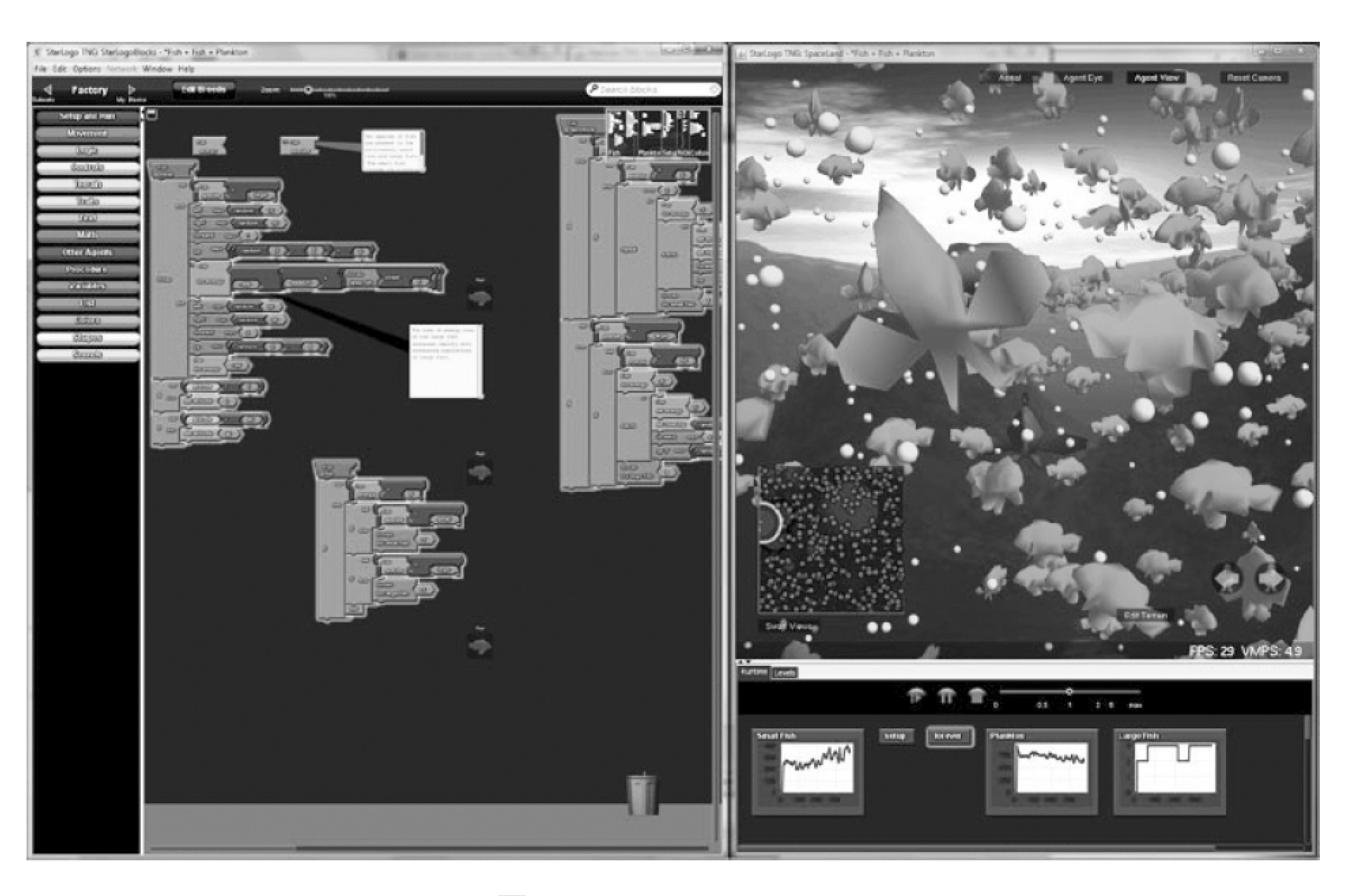


图 1.8　StarLogo TNG

Scratch 是麻省理工媒体实验室开发的一款面向儿童的简易编程工具，旨在通过游戏式的方式激发深层次的学习，如图 1.9 所示。用户可以利用 Scratch 创建互动动画、故事或游戏，并可通过网络与其他开发者分享自己的创造成果。Scratch 的学习可以为日后学习更高级别的编程语言奠定坚实的知识基础。MIT 的 Scratch 团队重视软件的易学性，创建和调试 Scratch 程序都非常简易。最早的 Scratch 版本于 2006 年由“终生幼儿园团队”(Lifelong Kindergarten Group)发布。

StarLogo TNG 和 Scratch 在很多方面对 App Inventor 产生了重要的影响，比如都采用拖曳的编辑方式、模块化的编辑语言，且 App Inventor 和 Scratch 一样，都致力于为初学者创造更愉快和简易的编程体验。

两者的不同之处在于，App Inventor 是一款用来开发智能手机程序的工具。因为智能手机在当今年轻群体中的流行和普及，App Inventor 拥有极大的潜力来吸引越来越多的年轻人从事软件开发工作，并运用计算思维分析和解决问题。



图 1.9　Scratch

1.3　App Inventor 的优势与不足

目前比较流行的 Android 开发方式是使用 Eclipse 编写 Java 代码，Eclipse 集成开发环境如图 1.10 所示。使用代码进行程序开发是目前较为成熟且普遍的方法，这种开发方式对开发人员的开发知识和经验具有一定的要求，对于刚刚接触程序开发或者没有程序开发经历的用户来说，使用代码开发是一件较为困难的事情。

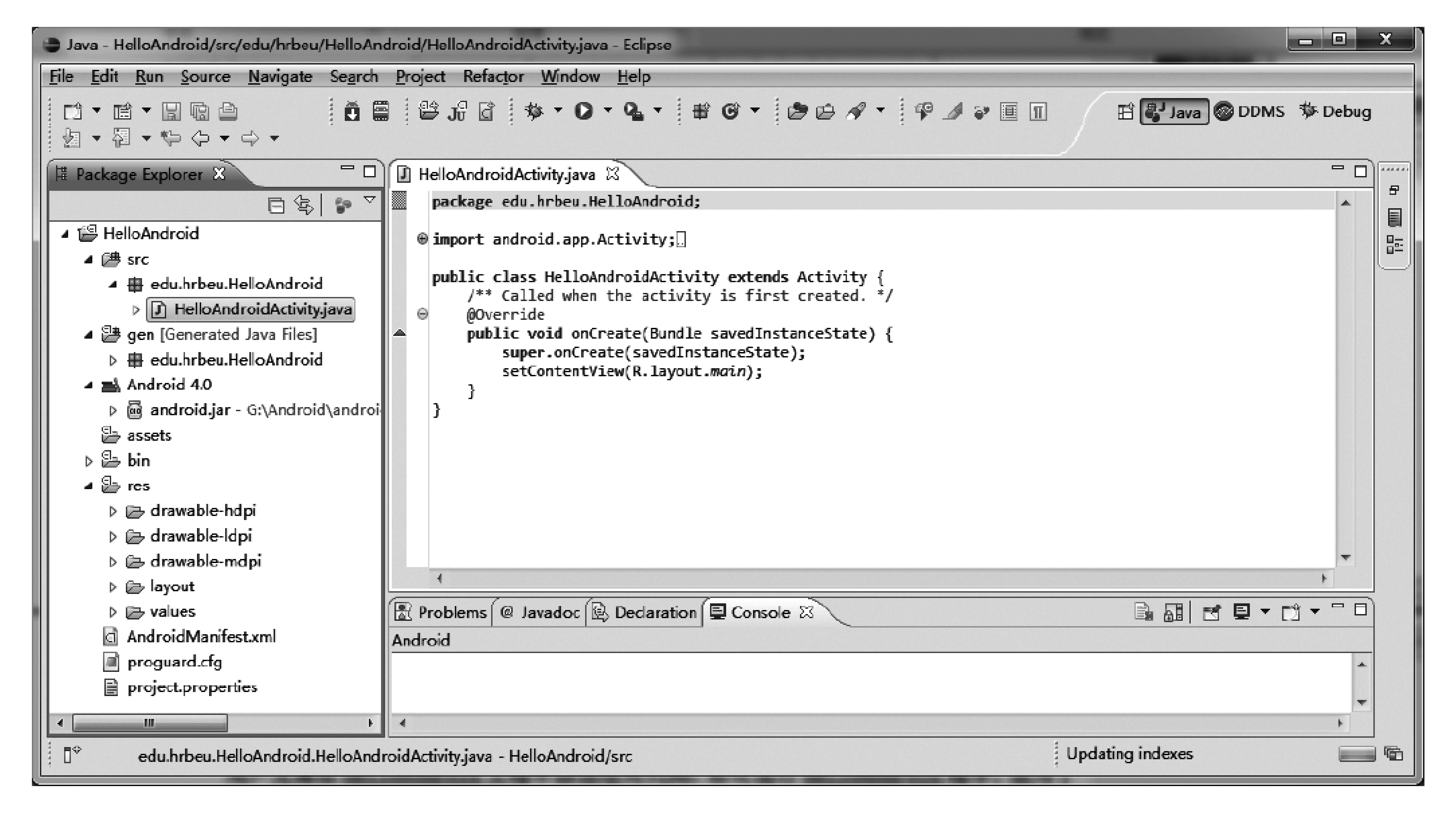


图 1.10　Eclipse 集成开发环境

相比之下，App Inventor 为用户提供了更为便捷的开发环境和方法，具有操作简单、可视化、模块化、事件置顶、正确性高和便于调试等优点。

1. 操作简单

使用 App Inventor 无须具备编程知识，也不需要记忆和编写代码，程序的组件和功能都存储在模块编辑库中，在创建程序时只需将其拖曳到编辑区域进行组合即可，用户不需要记忆如何输入指令或参考任何编程设计手册，如图 1.11 所示。

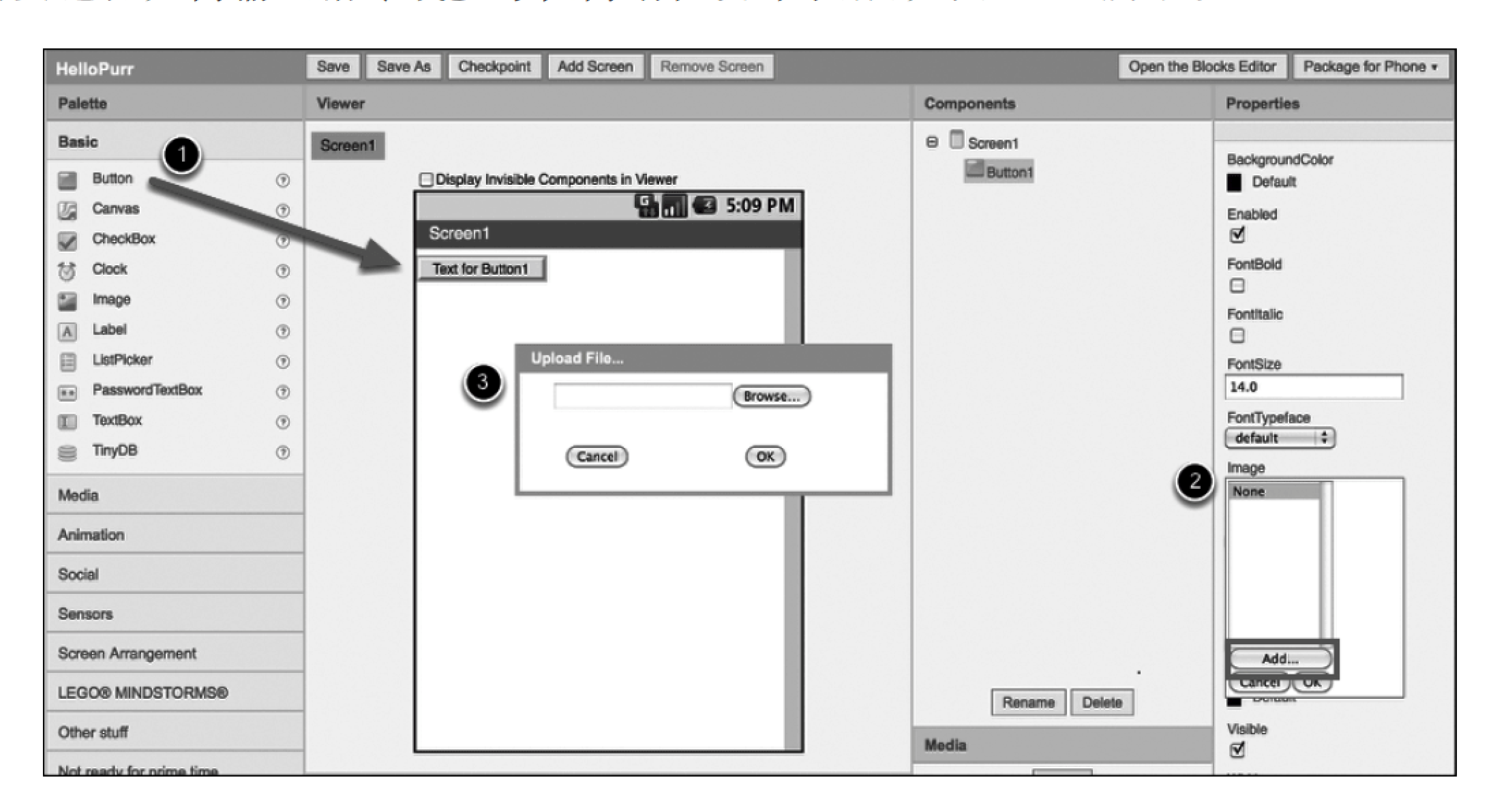


图 1.11　界面编辑器拖曳组件完成界面开发

2. 可视化和模块化

在 App Inventor 中，不仅用户界面开发是可视化和模块化的，程序逻辑的开发也是如此。在图 1.12 中，模块被分为不同的类别，并且标记成不同的颜色，执行不同的动作。在设置每个组件的行为时，犹如玩乐高积木，逻辑关系一目了然。

3. 事件处理器

在传统的编辑语言中，对程序最贴切的比喻是"一个程序就像一个处方，一个说明书"。然而，随着图形用户界面的出现，程序不再像处方一样了，而变成了"事件处理器"。如图 1.13 所示，单击按钮 Button 1 时，程序播放音频 sound 1，这便是正确的事件处理器概念模型。

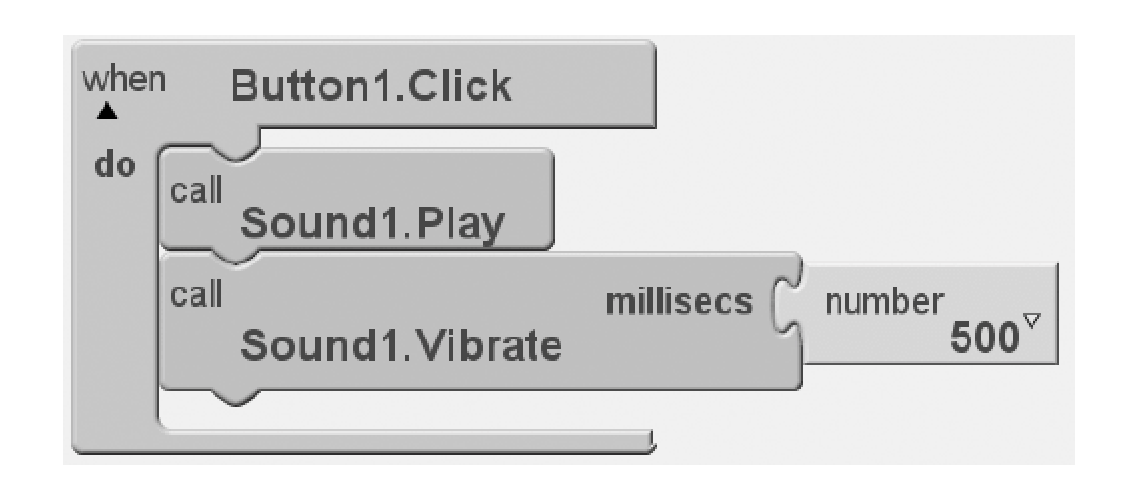


图 1.12　App Inventor 模块化编辑语言



图 1.13　事件置顶

对于 App Inventor 来说,一个应用程序便是一套事件处理器。当用户想要设计一个按钮被单击后的效果时,只需拖曳出按钮的单击事件模块,并把单击后要发生的行为模块放置在按钮单击事件模块中就可以了。在设计应用软件的过程中,模块的每个功能行为都预先设计好,并摆放在开发环境中供用户使用,这样不但简化了程序开发工作,也使整个编程过程显得分外清晰。

4. 正确性高且便于调试

在代码式编程过程中,出现错误后信息比较隐秘,无法简单地遏制错误的发生。而 App Inventor 的模块编辑语言可以从一开始就限制编程的出错几率。例如,如果选择了一种参数模块槽,便无法将其他类型的参数模块与其拼接,这样便降低了参数设置错误的几率。App Inventor 允许相匹配的模块进行拼接,这个特点在一定程度上保证了编程的正确性。

如果编程过程中出现了错误,可以利用 App Inventor 的回收站,用户只要将错误的组件直接拖曳进去便可删除,这比起代码开发方式中对错误的修补要方便简捷得多。在应用程序的开发过程中,用户可以随时在自己的 Android 设备上或模拟器上进行调试,发现的错误可以随时进行修改。

当然,App Inventor 也有它的不足之处。目前 App Inventor 开发出来的程序,只能通过下载安装在手机上,或者下载到计算机上,但不能发布到 Google Play 上供他人下载。App Inventor 目前还不能产生 Java 代码,因此也无法将开发完成后的程序导入 Eclipse 进行再次开发。上述 App Inventor 的不足之处,谷歌官方正在积极解决,希望不久的将来 App Inventor 可以更加成熟、易用。

1.4 App Inventor 作品展示

本节介绍使用 App Inventor 制作出来的手机软件作品,以便使读者能够直观地了解 App Inventor 的开发能力。

如图 1.14 所示,这是一款增扩实境的国际象棋游戏。玩家用手机选择自己的团队



图 1.14 增扩实境国际象棋游戏

和开棋的位置，利用室外的开阔地作为棋盘，然后他们充当棋子来回移动。游戏通过GPS来记录每个玩家的移动轨迹，并在手机屏幕上展示玩家在虚拟棋盘上的位置。

美国海军陆战队上士Chris Mstzger，利用App Inventor开发了一款应用软件，可以帮助海军陆战队士兵摧毁在战场上发现的炸药，如图1.15所示。

在海地，人道主义开源软件项目利用App Inventor开发了两款软件，帮助了那里的人道主义救援人员实地记录降雨量和物价的变化，如图1.16所示。



图1.15　弹药检查软件



图1.16　统计软件

阿拉巴马州劳伦斯郡高中的学生用App Inventor开发了一款物种检查软件，用来记录野猪的出没，如图1.17所示。这款软件所记录的数据，将帮助科学家了解野猪入侵的问题。

Google图书搜索软件在搜索时，用户可以输入书籍的全名或书名的关键词，然后根据用户所输入的内容显示最相关的书籍信息，图1.18是该软件的运行界面。



图1.17　物种检查软件

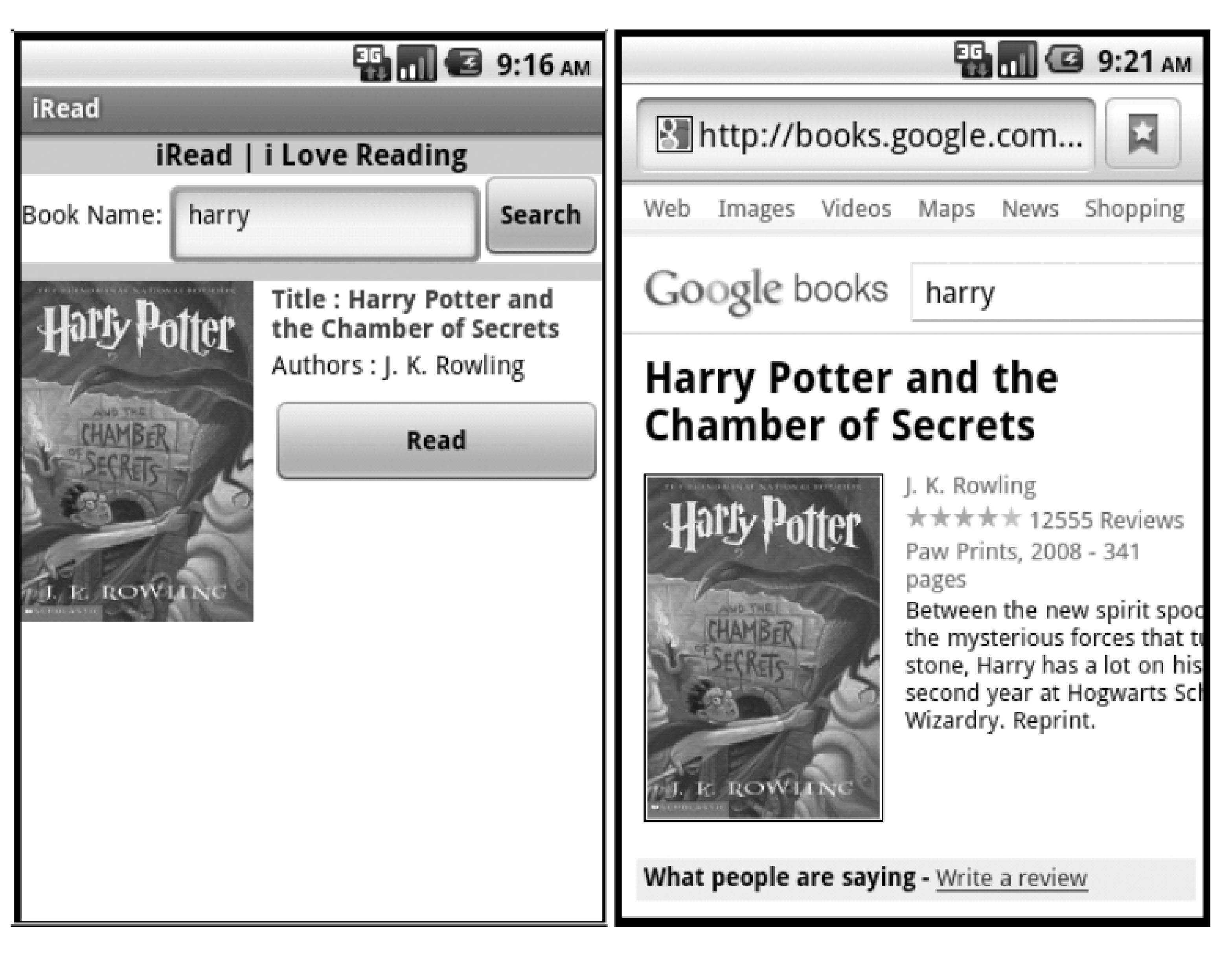


图1.18　Google图书搜索软件

1.5 App Inventor 学习资源

通过前面的介绍，相信读者已经对 App Inventor 有了一定的了解，下面介绍一些 App Inventor 的学习资源网站，这些网站中的资源可以帮助读者更好地学习和使用 App Inventor，创造出更多有趣的应用程序。

1. MIT App Inventor

MIT App Inventor(http://appinventor.mit.edu)为学习者提供了由浅入深的软件制作课程，如图 1.19 所示。其中的 Teach 部分为教学提供了文章、书籍和教程等多种教学辅助资源；Explore 部分展现了 App Inventor 的开发能力、作品和开发团队等内容；Invent 部分可以使用 App Inventor 进行 Android 应用程序开发。



图 1.19　MIT App Inventor 网站

2. App Inventor 中文学习网

该网站(http://www.appinventor.tw/home)为 CAVE 教育国际与翰尼斯企业有限公司合作架构的 App Inventor 教育平台，为学习者提供优秀的网络学习环境、中文说明和一些小应用程序的源代码，如图 1.20 所示。

3. App Inventor Blocks

该网站(http://www.appinventorblocks.com/home)介绍了如何安装和配置 App Inventor，介绍了一些界面组件的用途，并提供范例程序的源代码，如图 1.21 所示。该网站提供了一些 App Inventor 作品的展示，在该网站 Other Resources 栏目当中，可以找到网站推荐的一些学习资源。



图 1.20　App Inventor 中文学习网

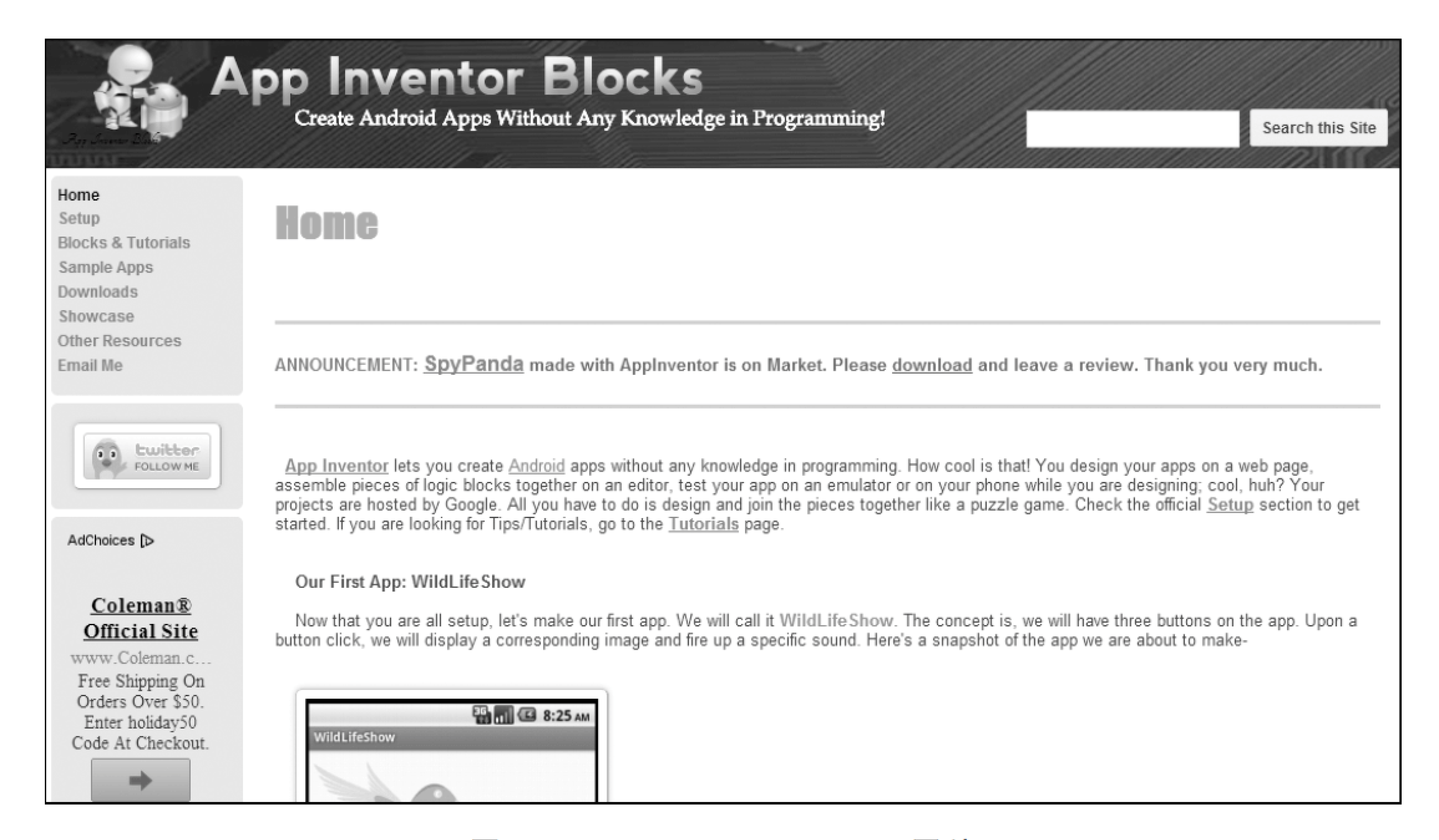


图 1.21　App Inventor Blocks 网站

4. 高师大附中资讯社 App Inventor 教学网

该网站(https://sites.google.com/a/stu.nknush.kh.edu.tw/appinventor/)属于台湾高师大附中资讯社，为学习者提供了全面的 App Inventor 背景和开发环境介绍，并配有教学范例，是为数不多的适合初学者的中文网站，如图 1.22 所示。

5. Stevozip

学习者可在该网站(https://sites.google.com/site/stevozip)中找到关于 App Inventor 的视频和非视频教程，以及技术论坛等资源，如图 1.23 所示。



图 1.22 高师大附中资讯社 App Inventor 教学网

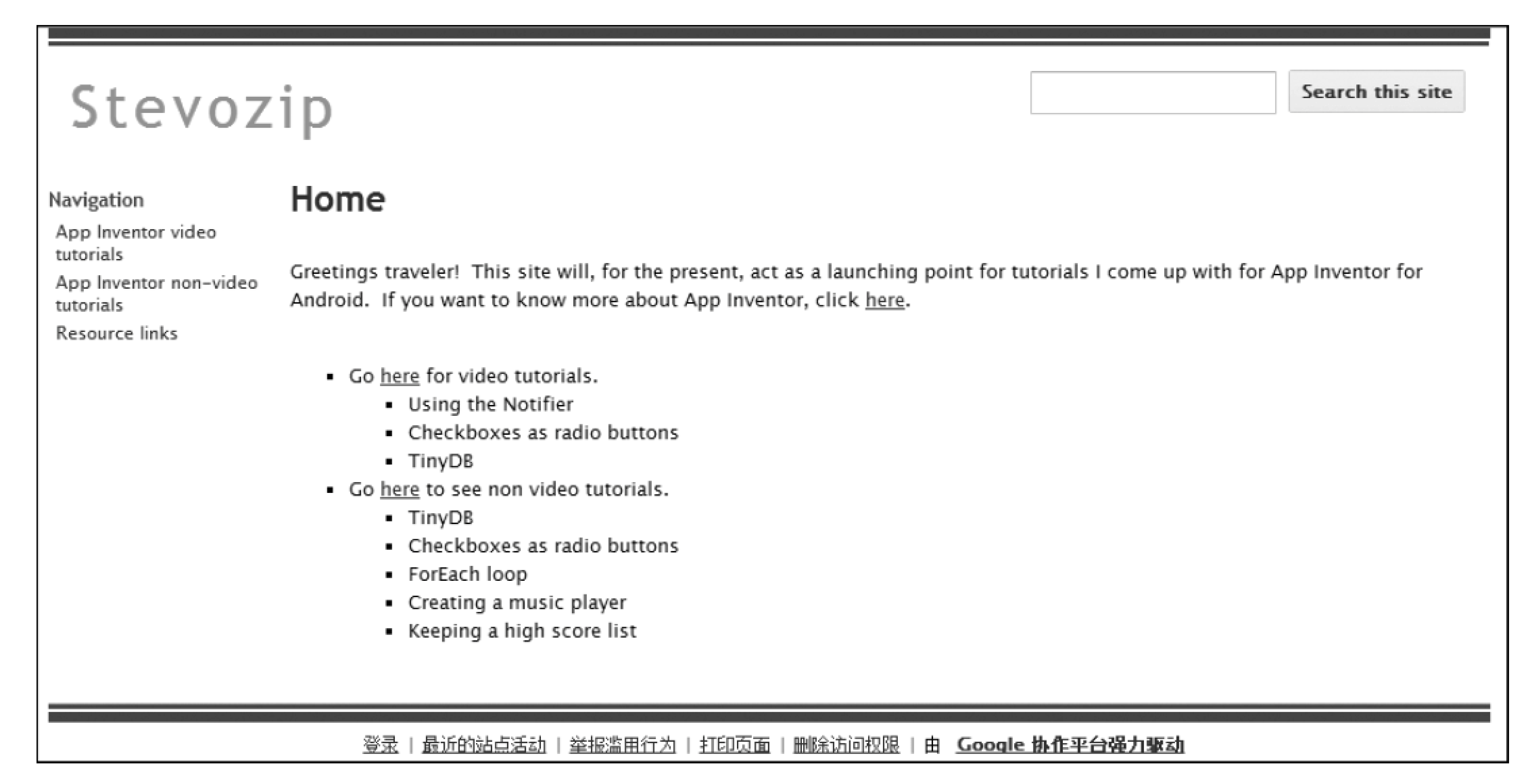


图 1.23 Stevozip 网站

除此之外,还有许多英文网站也提供了 App Inventor 的各种开发资源:

(1) tAIR——The App Inventor Repository(http://www.tair.info/)(图 1.24);

(2) Pura Vida Apps(http://puravidaapps.com/index.php);

(3) Android Aid(http://android.jwtyler.com)。



图 1.24 App Inventor 学习资源网站

习　　题

1. 简述 App Inventor 与传统编程方式的区别。
2. 简述 App Inventor 的优势和不足。
3. 尝试从 App Inventor 学习资源网站中，下载并运行一些小应用程序。

第 2 章



App Inventor 开发环境

App Inventor 开发环境的安装与配置是开发应用程序的第一步，也是深入理解 App Inventor 的一个良好的途径。通过本章的学习，读者可以掌握安装、配置 App Inventor 开发环境的步骤和注意事项，熟悉 App Inventor 的界面编辑器和模块编辑器，掌握使用手机和模拟器进行程序调试的方法。

本章学习目标：

- 掌握 App Inventor 开发环境的安装配置方法；
- 熟悉 App Inventor 的界面编辑器和模块编辑器；
- 掌握使用手机和模拟器进行程序调试的方法。

21 开发前准备

使用 App Inventor 进行开发之前，首先要检查一下所使用的操作系统和浏览器是否支持 App Inventor 开发。App Inventor 所支持的操作系统和浏览器如表 2.1 和表 2.2 所示。

表 2.1 App Inventor 所支持的操作系统

操作系统	版本说明
Macintosh	Mac OS X 10.5 或更高版本
Windows	Windows XP，Windows Vista，Windows 7
GNU/Linux	Ubuntu 8 或更高版本，Debian 5 或更高版本

表 2.2 App Inventor 所支持的浏览器

浏览器名称	版本说明	浏览器名称	版本说明
Mozilla Firefox	3.6 或更高版本	Google Chrome	4.0 或更高版本
Apple Safari	5.0 或更高版本	Microsoft Internet Explorer	7 或更高版本

使用 App Inventor 开发前需要做 4 个步骤的准备工作，如图 2.1 所示。首先要安装 Java 运行环境，其次是安装 App Inventor 软件，然后是启动 App Inventor 进行 Android

程序开发，最后是使用 Android 设备或模拟器运行 Android 程序。

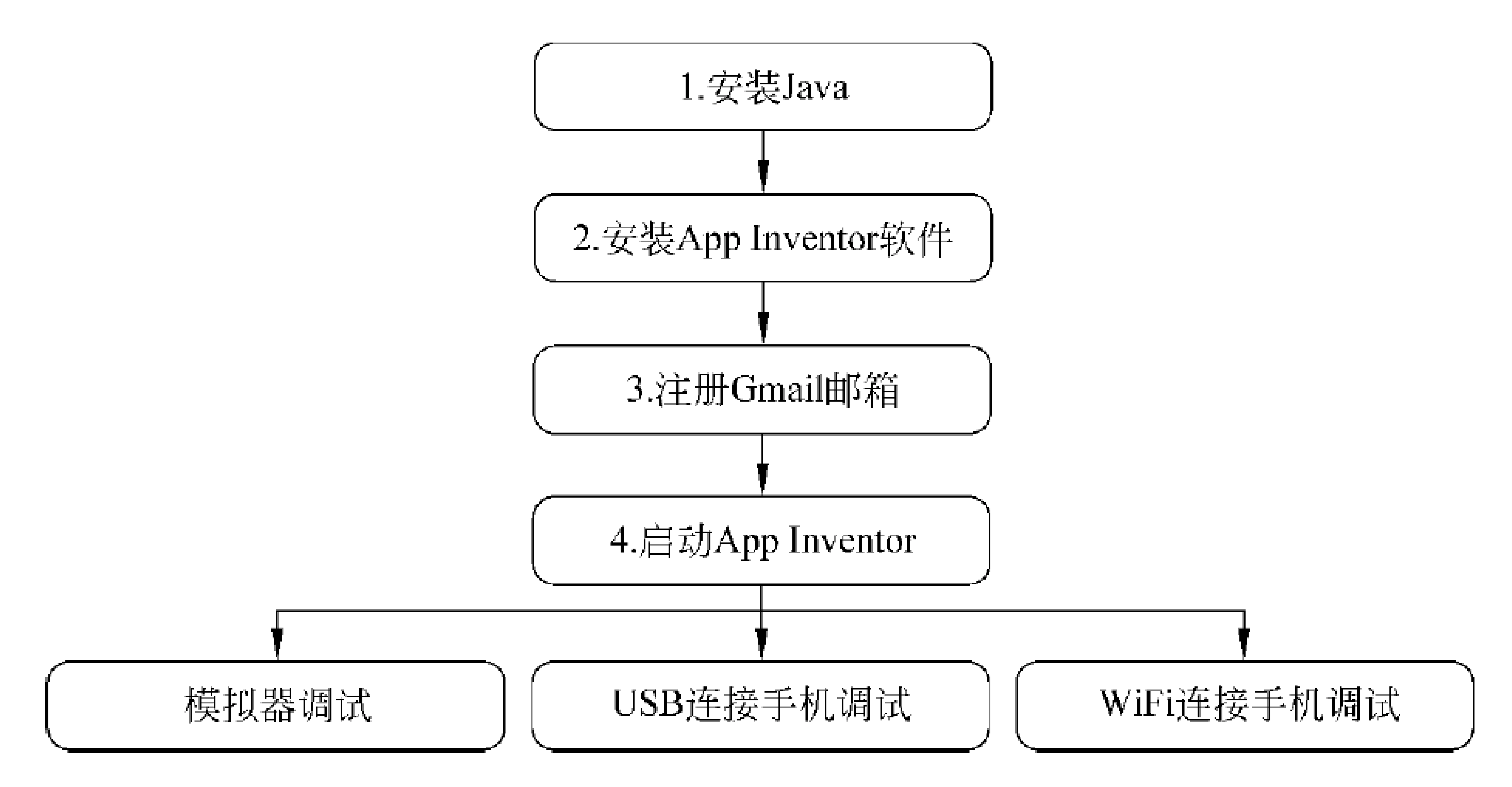


图 2.1　App Inventor 开发环境配置步骤

如果计算机操作系统中已经安装了 Java 运行环境，则可以直接进行第二步“安装 App Inventor 软件”，如果没有，请参考下面的 Java 运行环境安装方法。

首先在浏览器地址栏中输入网址“http://www.java.com”，进入 Java 下载首页，如图 2.2 所示。单击页面中的“免费 Java 下载”按钮跳转至最新版本的 Java 下载页面。



图 2.2　Java 下载首页

在最新版本的 Java 下载页面中，单击“同意并开始免费下载”按钮，开始下载最新版本的 Java 运行环境，如图 2.3 所示。笔者下载 Java 运行环境的版本是 7.11，文件名为 JavaSetup7u11.exe。

如图 2.3 所示的页面中没有正确识别读者使用的操作系统，则可单击页面下方的



图 2.3　最新版本的 Java 下载页面

“查看所有 Java 下载”链接，根据读者操作系统的类型选择适合的 Java 运行环境版本，目前 Java 有 Windows 版本、Mac OS X 版本、Linux 版本和 Solaris 版本。

下面以 Windows 系统为例介绍 Java 运行环境的安装和配置过程。首先运行笔者下载的 Java 运行环境文件 JavaSetup7u11.exe，启动安装程序，如图 2.4 所示。



图 2.4　Java 安装页面

如果无须修改 Java 运行环境的安装目录，则可以直接单击“安装”按钮，否则选择“更改目标文件夹”复选框，修改 Java 运行环境的安装目录。

Java 运行环境安装成功后，出现“您已成功安装 Java”的提示，如图 2.5 所示。

在进行下一步准备工作前，读者需要确认所使用的操作系统和浏览器是否被 App Inventor 支持。



图 2.5　Java 安装成功

22　安装 App Inventor 软件

安装 App Inventor 前，先要下载 App Inventor 软件包，在浏览器地址栏中输入网址“http://appinventor.mit.edu/explore/install-app-inventor-software.html”，进入 App Inventor 操作系统选择页面，如图 2.6 所示。

Install the App Inventor Setup Software

Before you can use App Inventor, you need to install some software on your computer. The software you need is provided in a package called *App Inventor Setup*. Follow the instructions for your operating system to do the installation, and then come back to this page to move on to Step 3 (starting App Inventor) and Step 4 (setting up your device or emulator).

- Instructions for Mac OS X
- Instructions for GNU/Linux
- Instructions for Windows

图 2.6　App Inventor 操作系统选择页面

App Inventor 支持 Mac OS X、GNU/Linux 和 Windows 三种操作系统。笔者使用的是 Windows 操作系统，因此选择 Instructions for Windows 进入 App Inventor 下载页面，如图 2.7 所示。

单击 Download 按钮下载 App Inventor 软件安装包，文件为 AppInventor_Setup_Installer_v_1_2.exe。双击下载的文件进入 App Inventor 安装界面，如图 2.8 所示。

单击 Next 按钮，显示 App Inventor 安装协议，如图 2.9 所示。

直接单击 I Agree 按钮，同意安装协议的条款，并进入 App Inventor 的目录选择界面，如图 2.10 所示。

如果没有特殊要求，可以使用 App Inventor 的默认目录进行安装。单击 Next 按钮，进入 App Inventor“开始”菜单目录选择界面，如图 2.11 所示。

Installing the App Inventor Setup software package

We recommend that you perform the installation from an account that has administrator privileges. This will install the software for all users of the machine. If you do not have administrator privileges, the installation should still work, but App Inventor will be usable only from the account you used when you installed.

1. Download the installer.
2. Locate the file **AppInventor_Setup_Installer_v_1_2.exe (~92 MB)** in your Downloads file or your Desktop. The location of the download on your computer depends on how your browser is configured.
3. Open the file.
4. Click through the steps of the installer. Do not change the installation location but record the installation directory, because you might need it to check the driver. The directory will differ depending on your version of Windows and whether or not you are logged in as an administrator.

图 2.7 App Inventor 下载页面

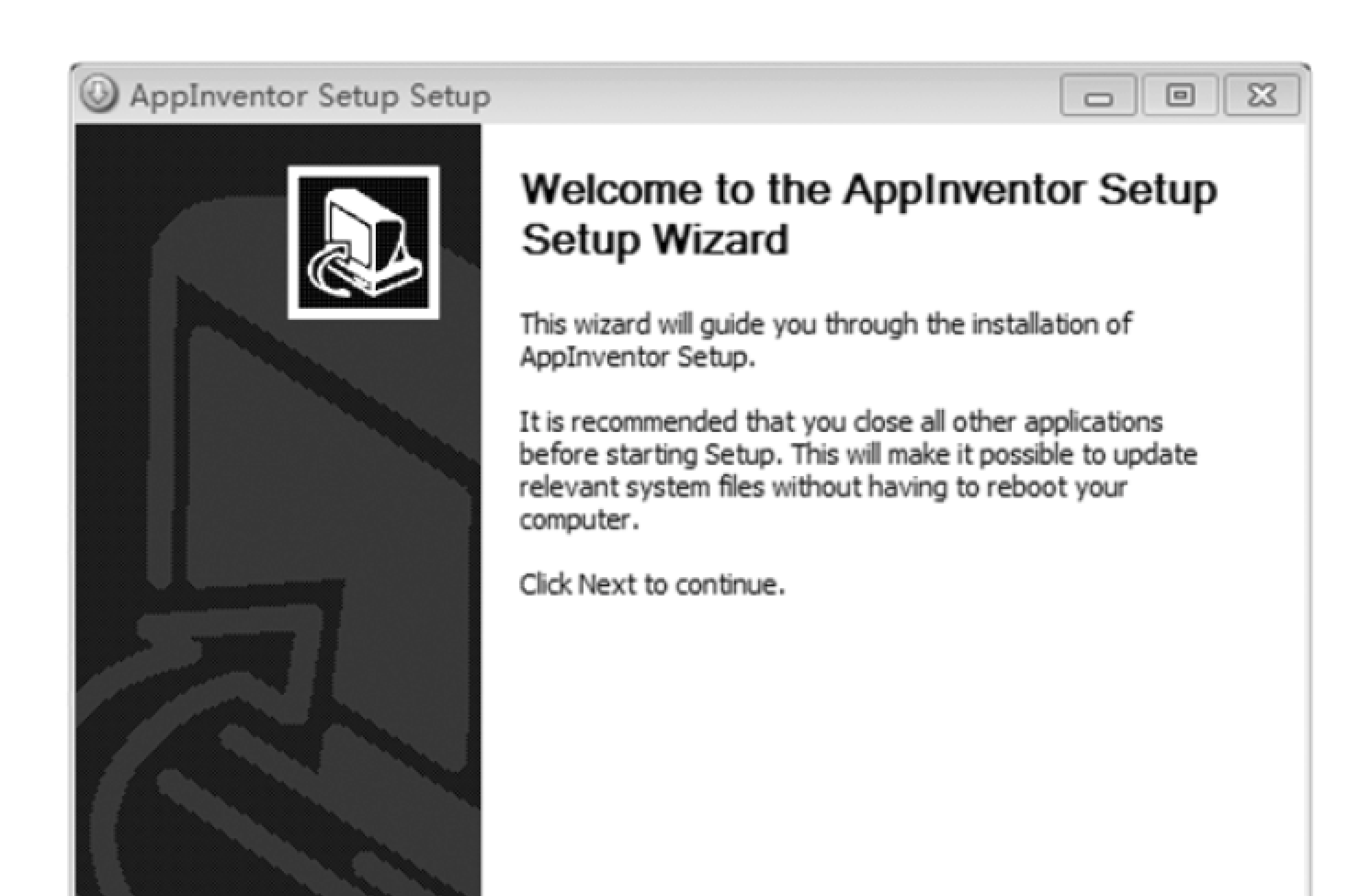


图 2.8 App Inventor 安装界面

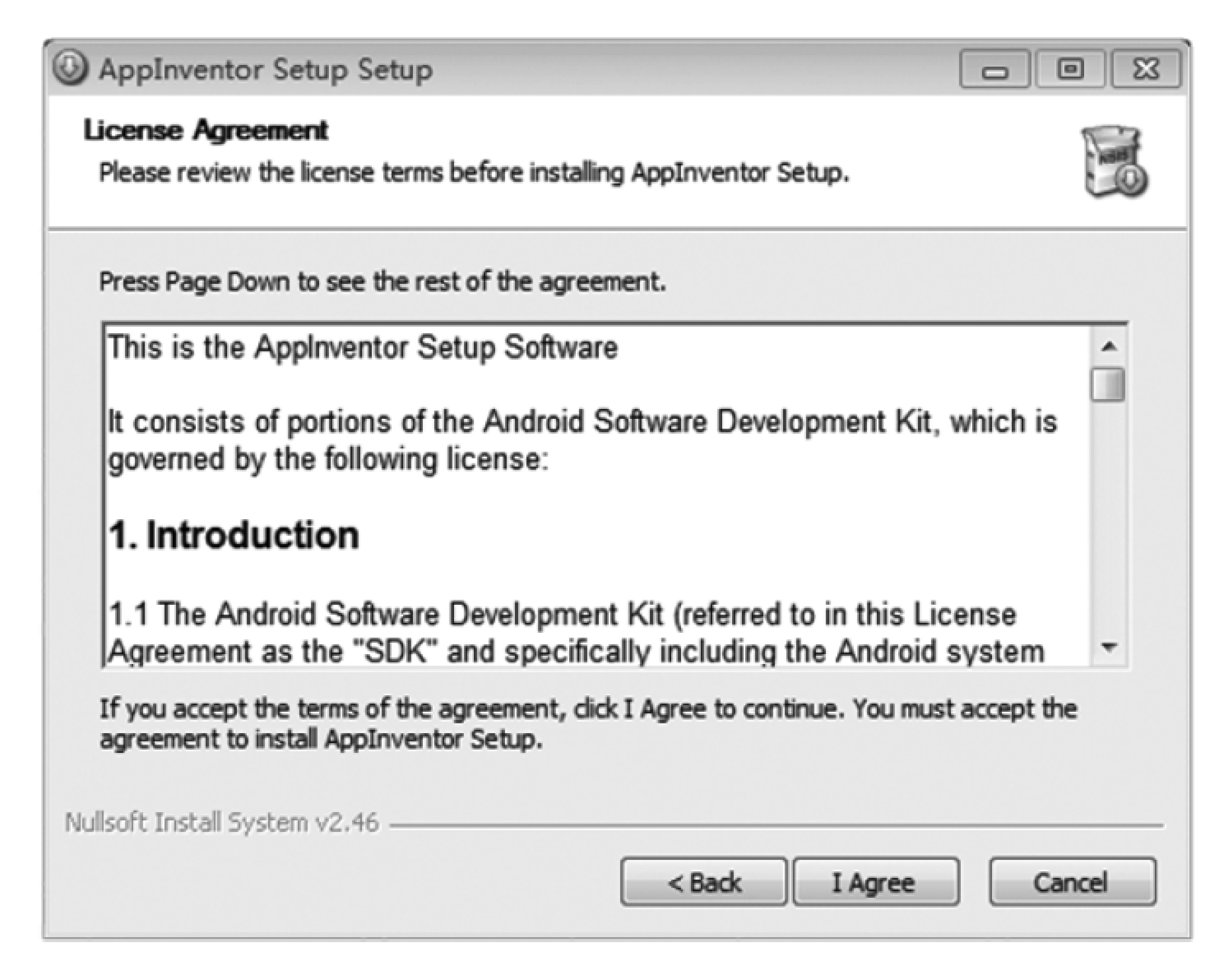


图 2.9 App Inventor 安装协议

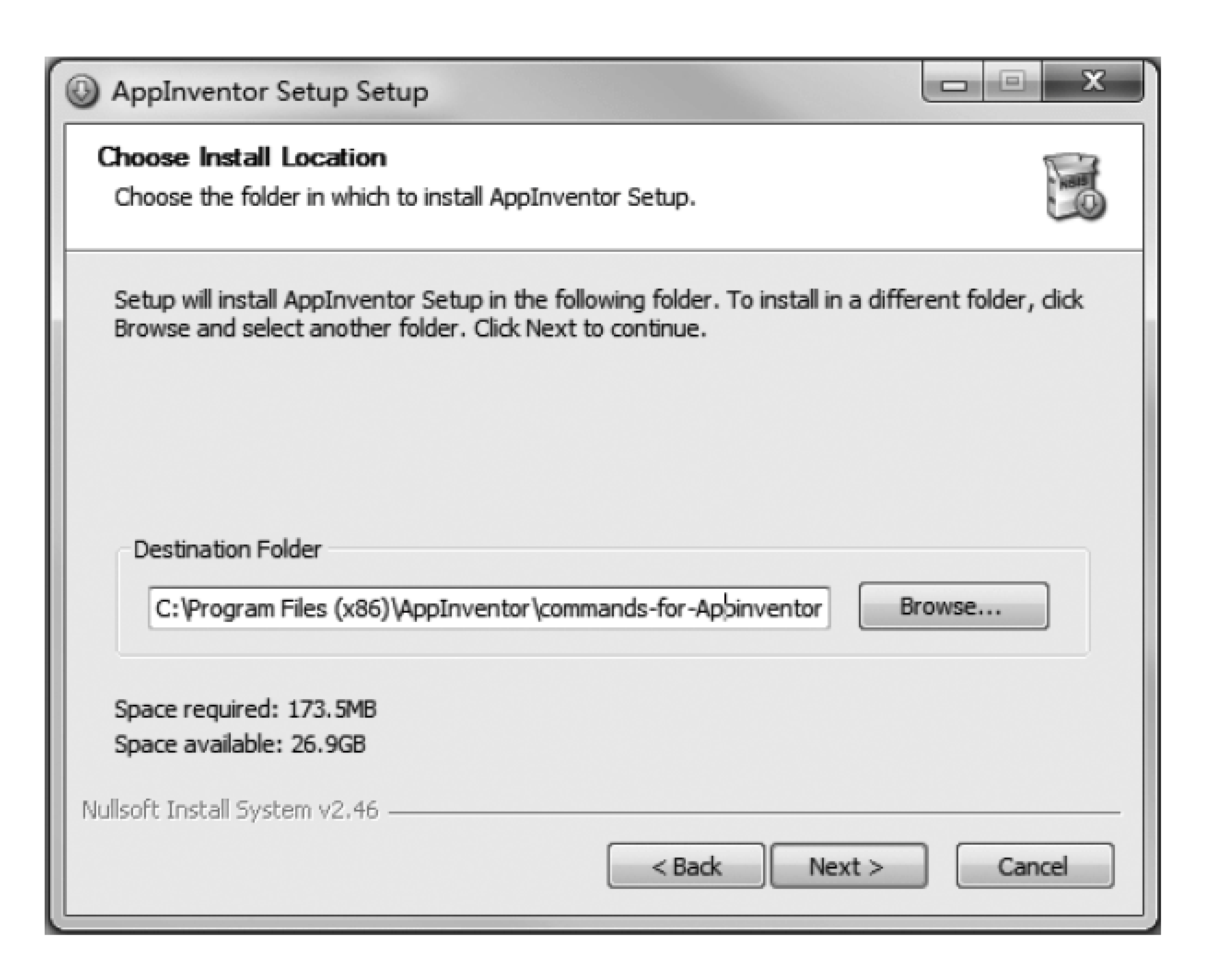


图 2.10　App Inventor 目录选择界面

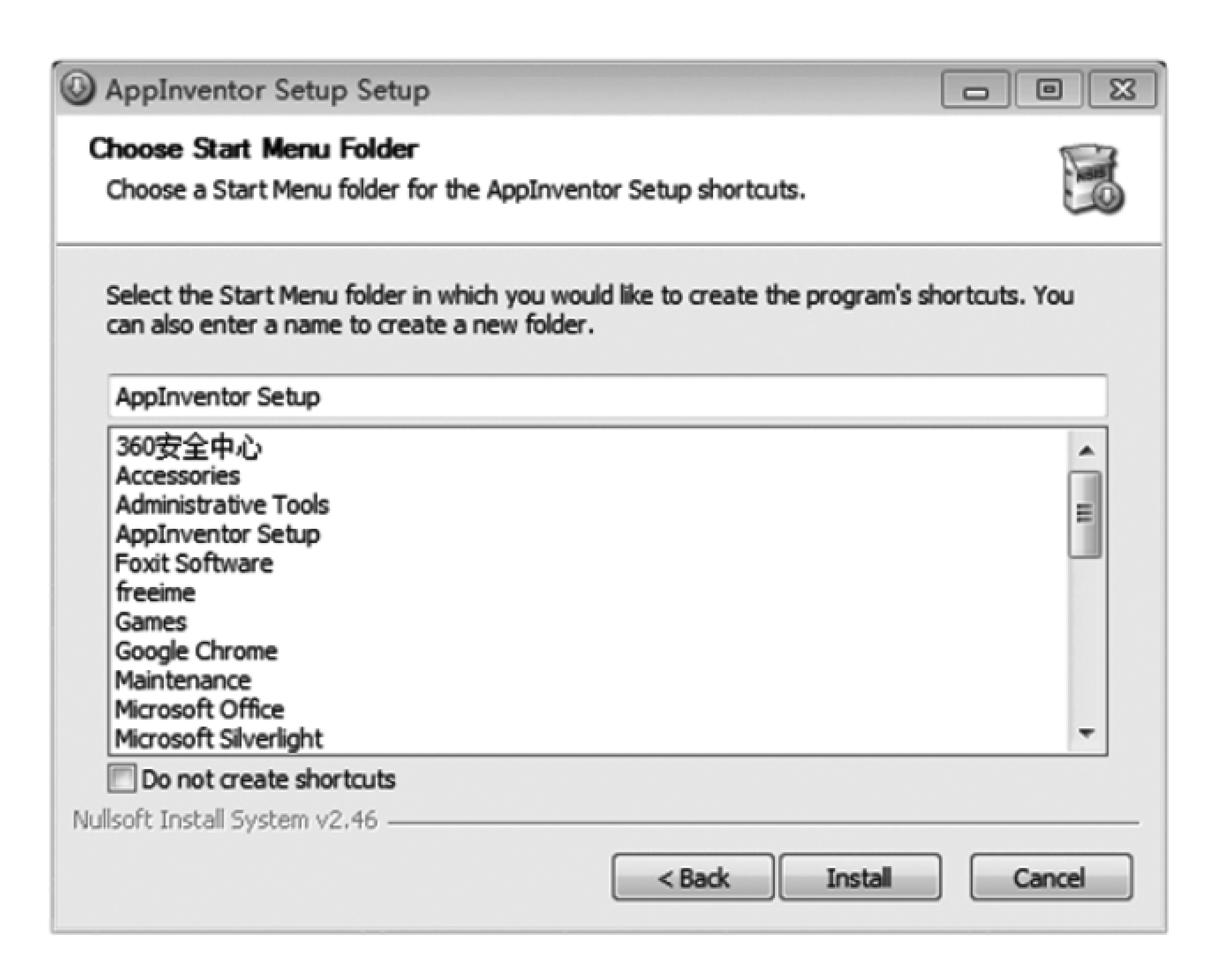


图 2.11　App Inventor“开始”菜单目录选择界面

不必修改“开始”菜单目录，直接使用 App Inventor 安装程序推荐的目录即可，然后单击 Install 按钮进行安装，如图 2.12 所示。

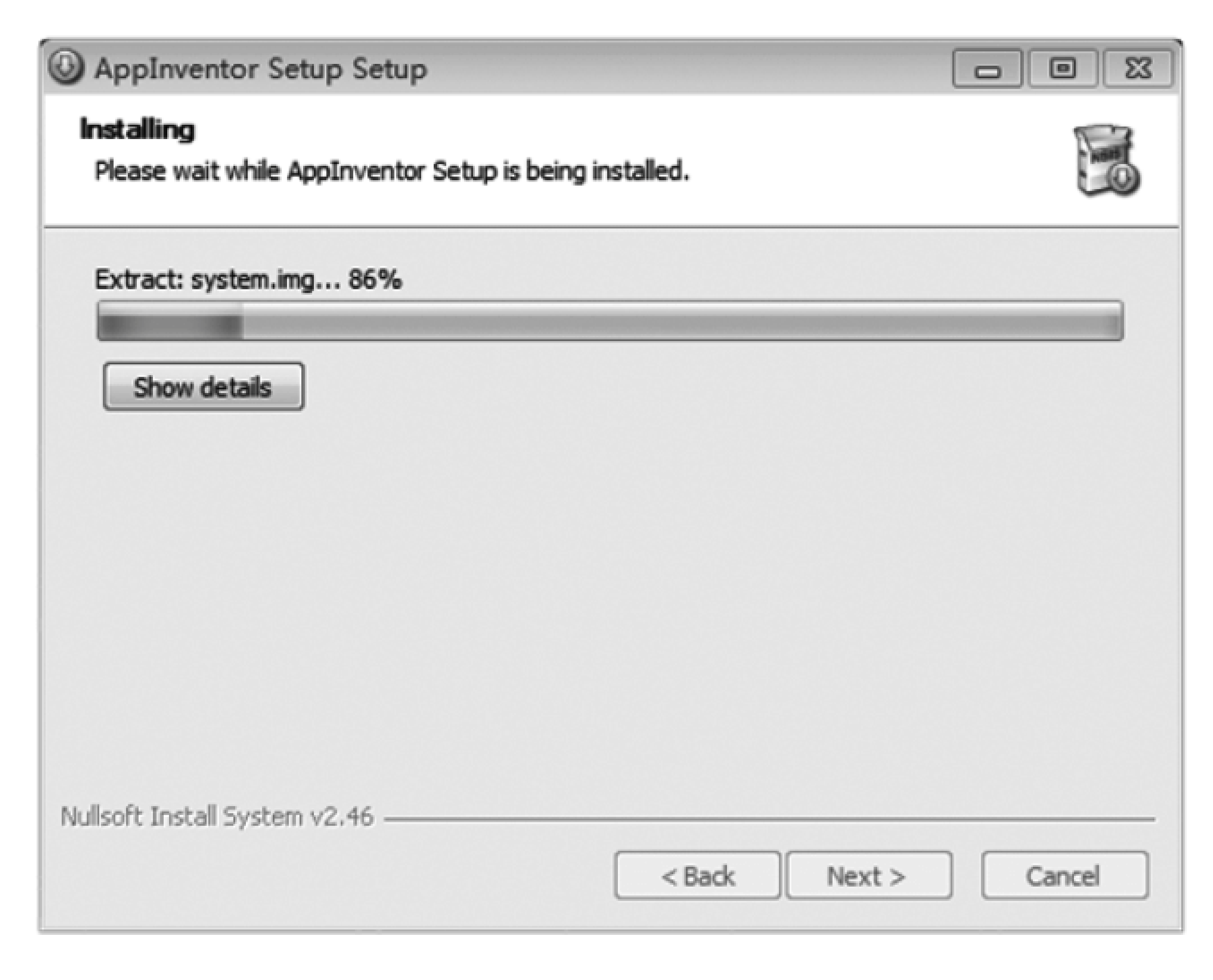


图 2.12　App Inventor 安装进度界面

安装完成后，会出现安装成功界面，如图 2.13 所示。单击 Finish 按钮退出 App Inventor 的安装程序。

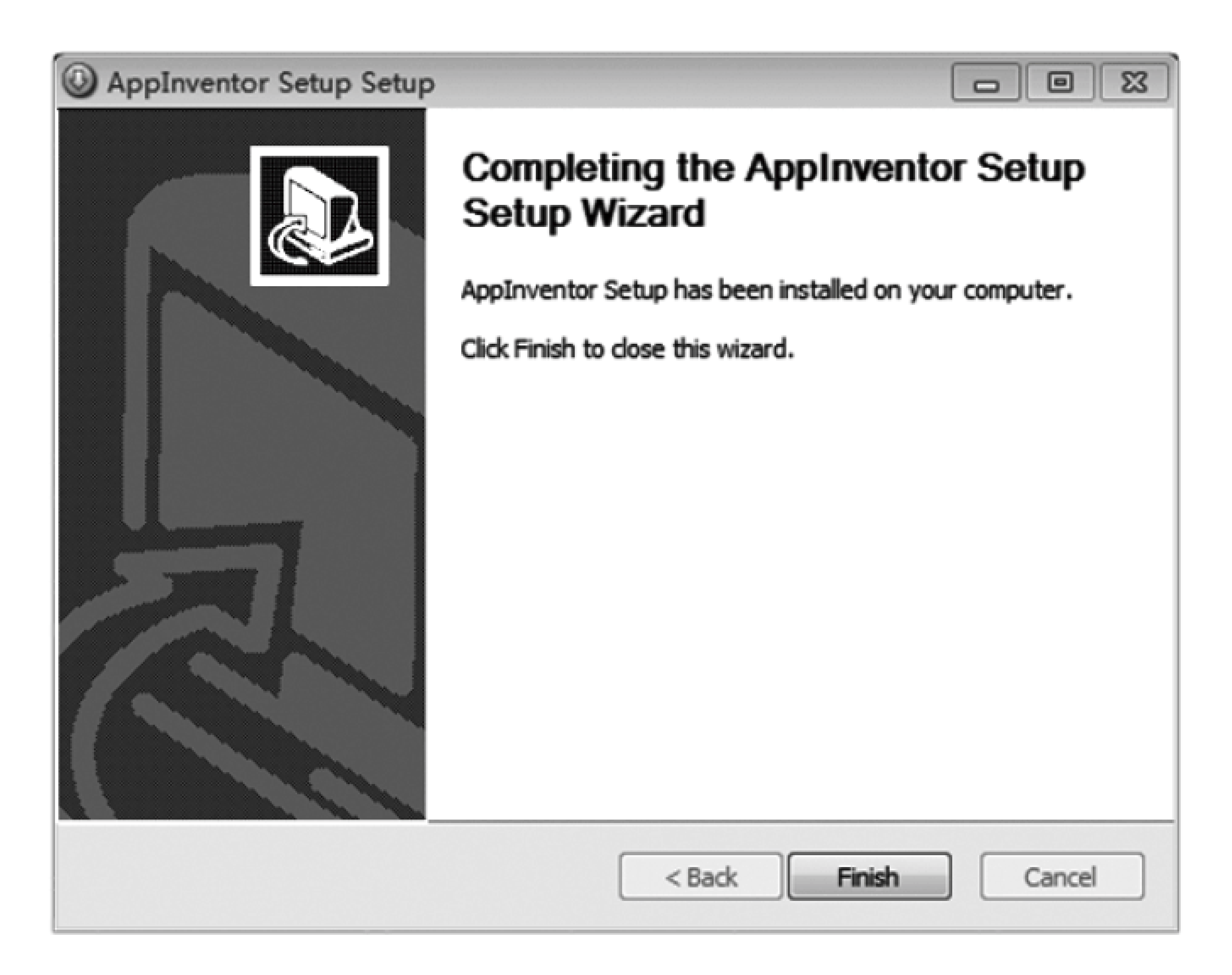


图 2.13　App Inventor 安装成功界面

2.3　注册 Gmail 邮箱账号

App Inventor 是一个基于浏览器的集成开发环境，为了能够区分不同的用户，App Inventor 需要使用 Google 的 Gmail 邮箱账号进行登录。因此在使用 App Inventor 之前，首先要介绍一下如何申请 Google 的 Gmail 邮箱账号。

打开 Web 浏览器，在地址栏中输入“http://www.google.com.hk”，进入谷歌搜索页面，如图 2.14 所示，单击页面右上角的“登录”按钮，进入 Gmail 邮箱登录页面，如图 2.15 所示。



图 2.14　谷歌搜索页面



图 2.15　Gmail 邮箱登录页面

单击页面右上角的“注册”按钮，进入 Gmail 邮箱注册页面，如图 2.16 所示。在 Gmail 注册页面中填写用户信息，完成账号注册过程。

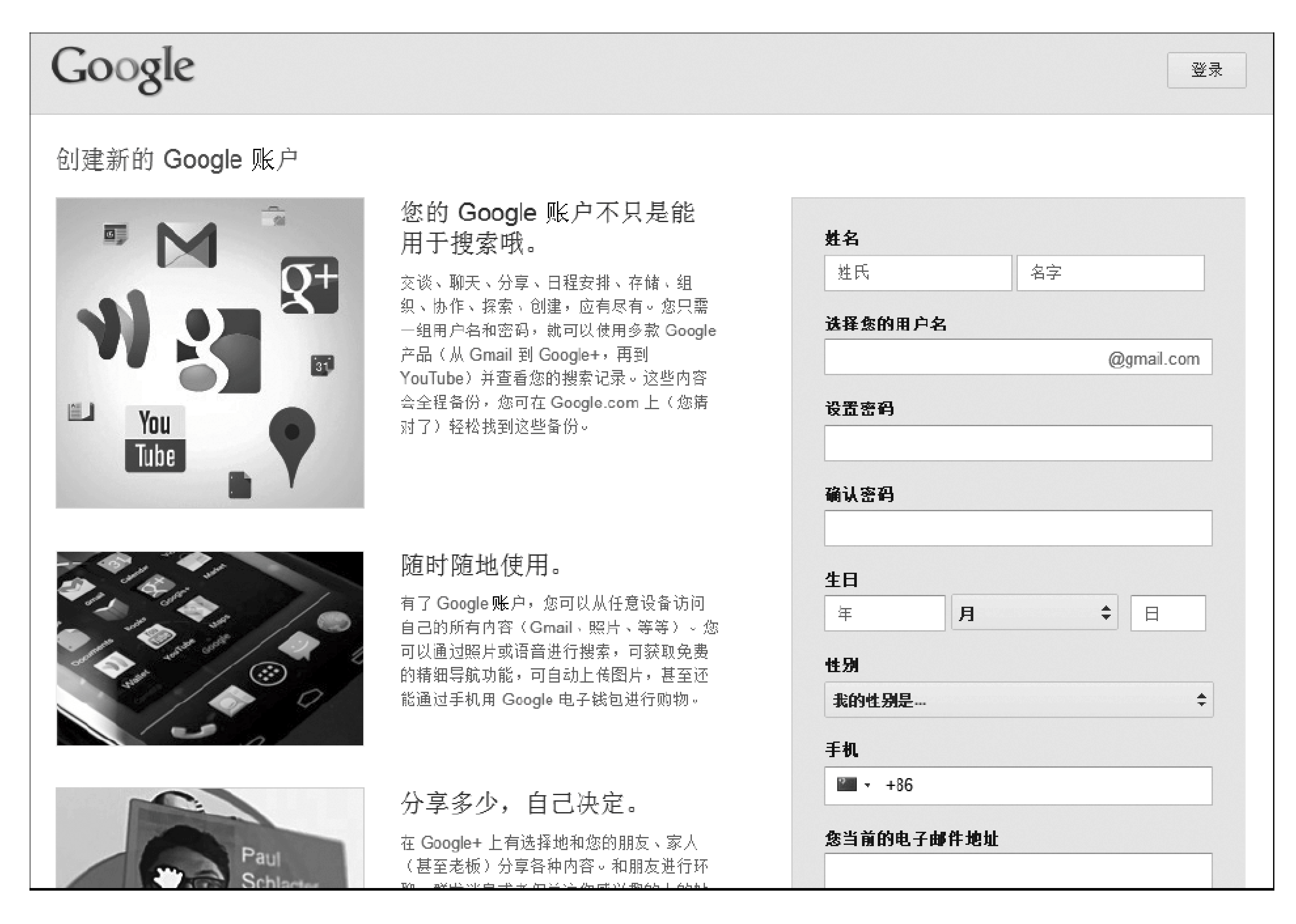


图 2.16　Gmail 邮箱注册页面

2.4　启动 App Inventor

首先进入 MIT App Inventor 的网站，如图 2.17 所示，单击 Invent 按钮，进入 Gmail 邮箱登录界面，使用先前注册的 Gmail 账号登录即可。

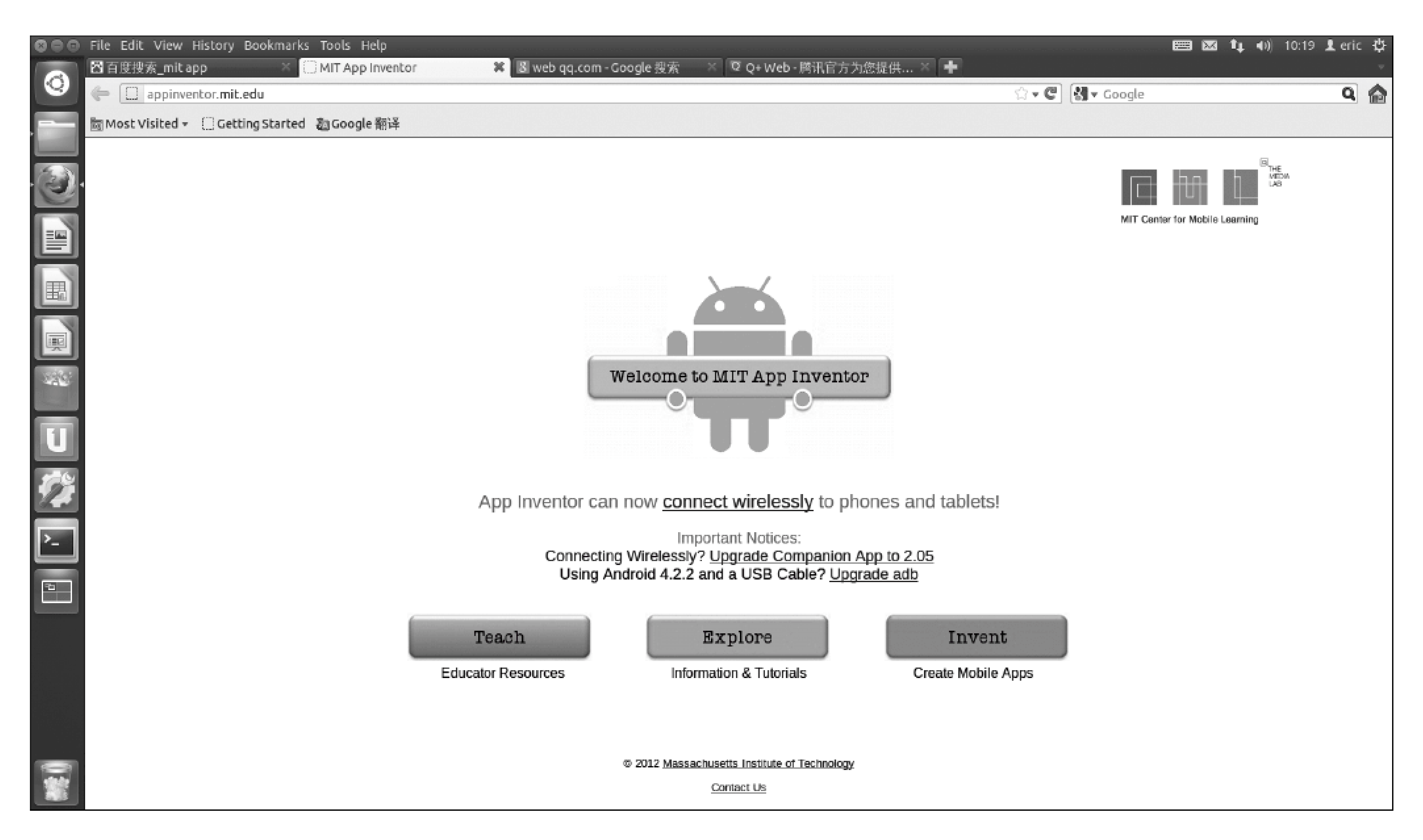


图 2.17　MIT App Inventor 网站

成功登录后，进入 App Inventor 的"我的项目"页面，如图 2.18 所示。"我的项目"页面，在用户首次登录时可以直接进入。首先简单介绍一下页面上的一些功能按钮，在图 2.18 中用数字标出的部分，将在下面逐一进行介绍。

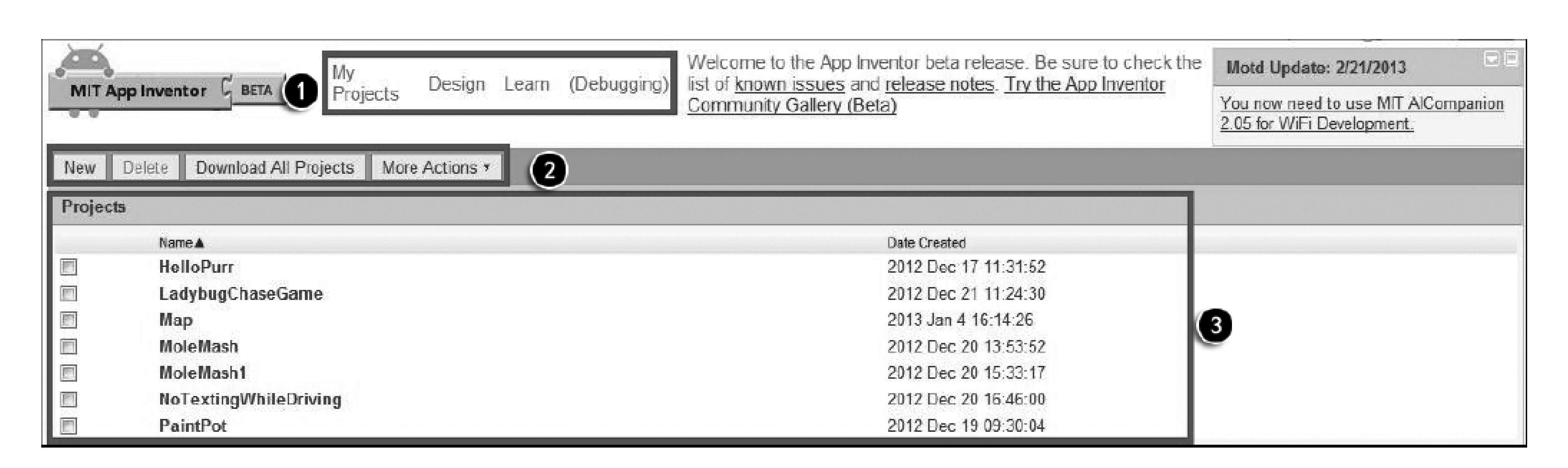


图 2.18　"我的项目"页面

(1) 我的项目(My Projects)、界面设计器(Design)和使用说明(Learn)链接。

My Projects 显示用户设计完成或正在进行的 Android 软件项目；Design 按钮可以打开"界面设计器"页面，"界面设计器"是辅助用户进行 Android 界面设计和开发的工具；Learn 按钮可以打开 MIT App Inventor 的学习资料页面，这里有各阶段的课程和教学资源。

(2) 新建项目(New)、删除项目(Delete)、下载全部项目(Download All Projects)和更多操作(More Actions)。

New 按钮可以新建一个工程项目；Delete 按钮则可删除已有的项目；Download All Projects 按钮可以将所有项目的源代码打包下载，只要在弹出的对话框中选择存储路径即可；More Actions 按钮支持更多操作，如下载某个项目的源代码(Download Source)、上传某个项目的源代码(Upload Source)、下载密钥(Download Keystore)、上传密钥

(Upload Keystore)和删除密钥(Delete Keystore)。

(3) 项目(Projects)列出了现有的所有项目，显示的内容包括项目的名称和创建时间。

2.4.1 界面设计器

界面设计器(Design)是 App Inventor 的重要组成部分，以可视化的方式设计 Android 程序的用户界面，提供了大量常用的界面组件，并可以设置组件的属性值，如图 2.19 所示。

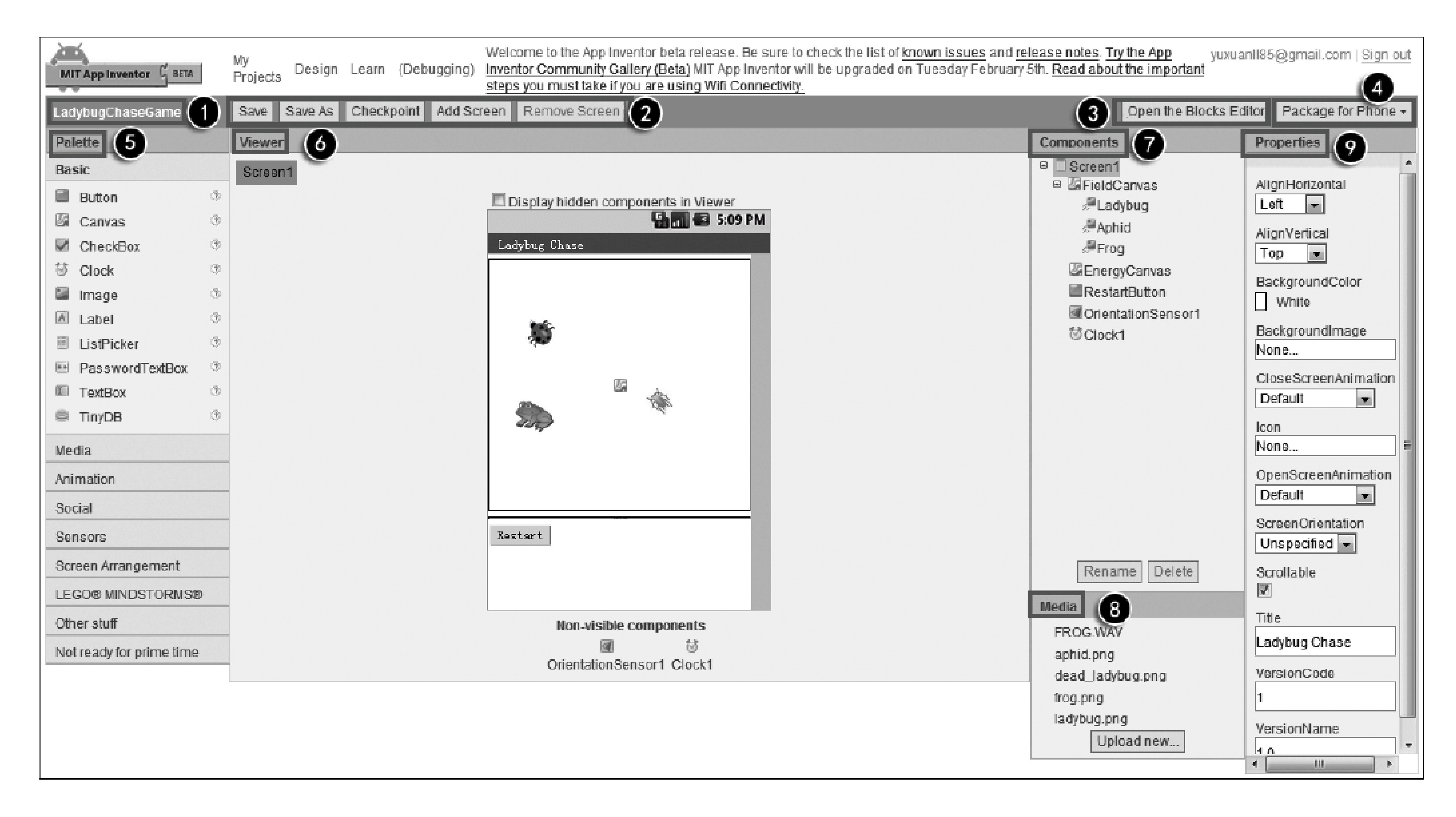


图 2.19 界面设计器

界面设计器主要由 9 个部分组成，下面依次介绍各部分的功能：

(1) 这里显示的是当前项目名称。

(2) 保存(Save)和另存为(Save as)用来保存当前项目；检查点(Checkpoint)用来设置回溯点，便于后期追溯和修改；增加屏幕页(Add Screen)和删除屏幕页(Remove Screen)用来增减屏幕页。

(3) 打开模块编辑器(Open the Blocks Editor)，当完成用户界面设计后，单击此按钮进入“模块编辑器”，进行程序逻辑方面的设置。

(4) 获取应用程序(Package for Phone)帮助用户将程序下载到计算机或手机中，有三种途径：显示二维码(Show Barcode)、下载到本机(Download to this Computer)和下载至手机(Download to Connected Phone)。其中，“显示二维码”将产生一个二维码图像，用户使用手机二维码扫描软件扫描此图像，则可以获得应用程序的下载链接地址，但这个链接只供开发者本人使用，因为需要开发者的 Gmail 账号登录后才可以下载。

(5) 控件库(Palette)，这里显示了 App Inventor 提供的全部界面控件。

(6) 设计区(Viewer)，用户在设计界面时，将“控件库”中的组件拖曳到设计区内，组

件将呈现在该区域内,并且组件的部分属性也会显示出来。

(7) 模块区(Components),以树形显示组件之间的关系,可以在此区域对组件进行重命名和删除。

(8) 资源区(Media),显示用户使用到的所有资源素材,如声音和图片。资源区下方的"上传新资源"(Upload new)按钮,可以将本地资源素材文件上传到资源区中。

(9) 属性设定区(Properties),可以在这一区域为每个组件设定所需的属性。

2.4.2 模块编辑器

模块编辑器(Blocks Editor)是Android程序逻辑和动作模块的设计区域。在界面设计器中单击Open the Blocks Editor按钮,下载并运行AppInventorForAndroidCodeblocks.jnlp文件,模块编辑器的界面如图2.20所示。

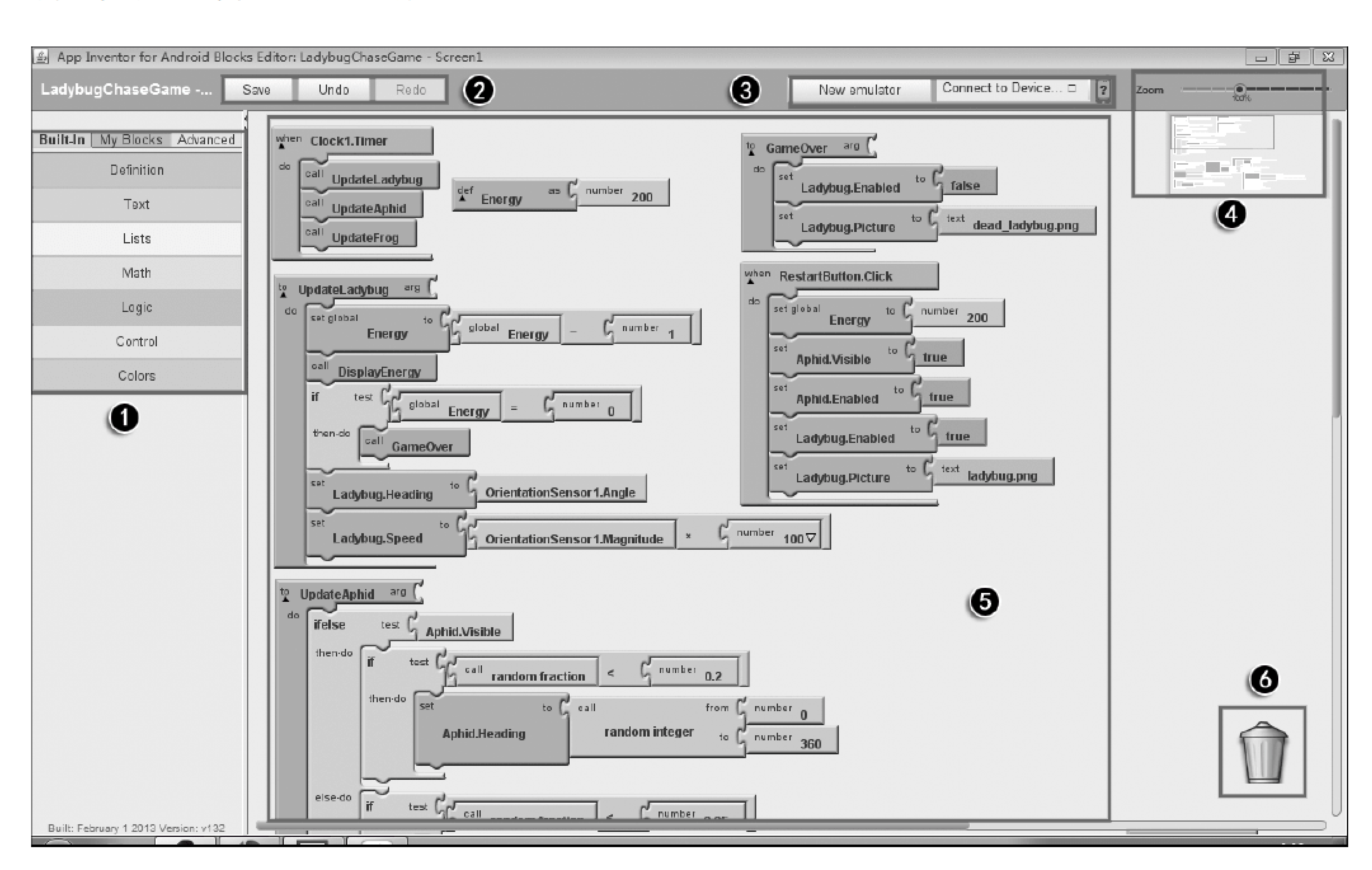


图2.20 模块编辑器

模块编辑器主要由6个部分组成,下面依次介绍各部分的功能。

1. 模块编辑库

在模块编辑库中可以找到进行程序逻辑设计的模块。开发者可以设定界面组件的逻辑关系和动作,这些都可以通过简单的拖曳和拼接就可以完成。

2. 保存(Save)、恢复上一步(Undo)和重做(Redo)

"保存"可以保存现有的模块拼接;"恢复上一步"是在本步骤操作出现错误时,重现上一步的内容;"重做"是重做本步骤内容。

3. 新建模拟器(New emulator)和连接设备(Connect to Device)

“新建模拟器”可以启动 Android 模拟器;“连接设备”的下拉列表中可选择已连接的手机或模拟器进行调试。

4. 设计区域缩放器

用于调整设计区域的大小,并可通过该控件观察到整个设计区域的布局。

5. 设计区

用于放置和编辑逻辑模块的区域,整个逻辑和动作的设计过程在此区域进行。

6. 回收站

用于删除逻辑模块,方法是直接将要删除的逻辑模块拖曳进回收站即可。

2.5 程序调试

App Inventor 提供多种程序调试方法,不仅可以将程序直接下载到模拟器中进行调试,还可以通过 USB 数据线或者 WiFi 连接,将程序下载到手机设备中进行调试。

2.5.1 Android 模拟器

Android 模拟器支持在没有手机设备的情况,在计算机上对 Android 程序进行开发、调试和仿真。目前,App Inventor 为用户提供的是 Android 2.2 版本的模拟器。

启动 Android 模拟器的方法是在模块编辑器的右上角单击 New emulator 按钮,打开模拟器启动提示界面,如图 2.21 所示。

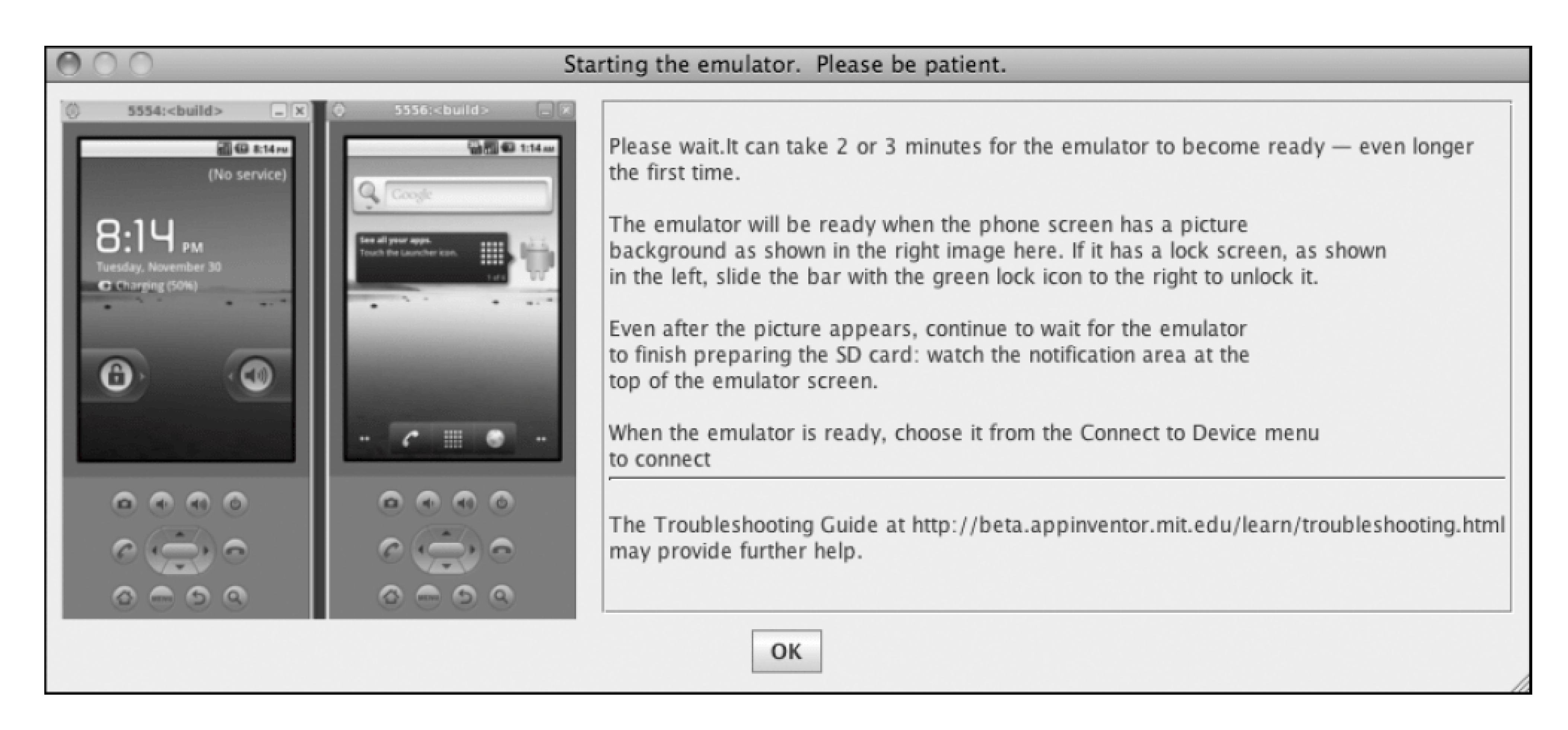
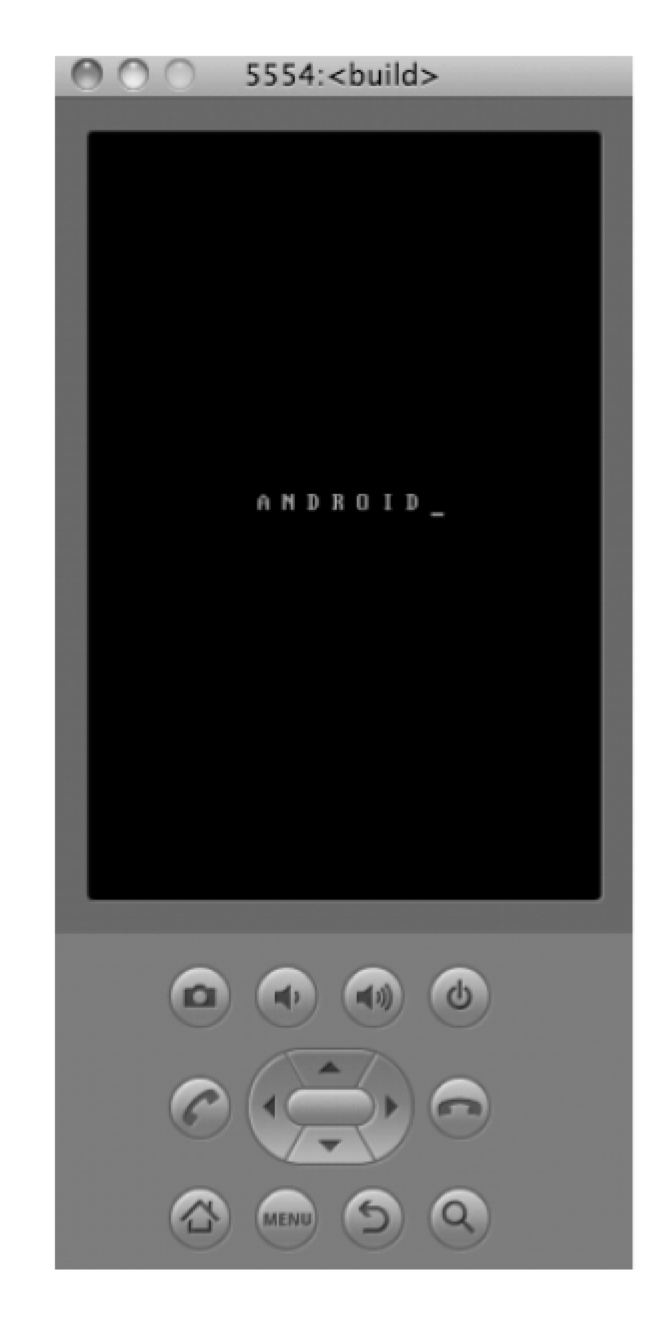
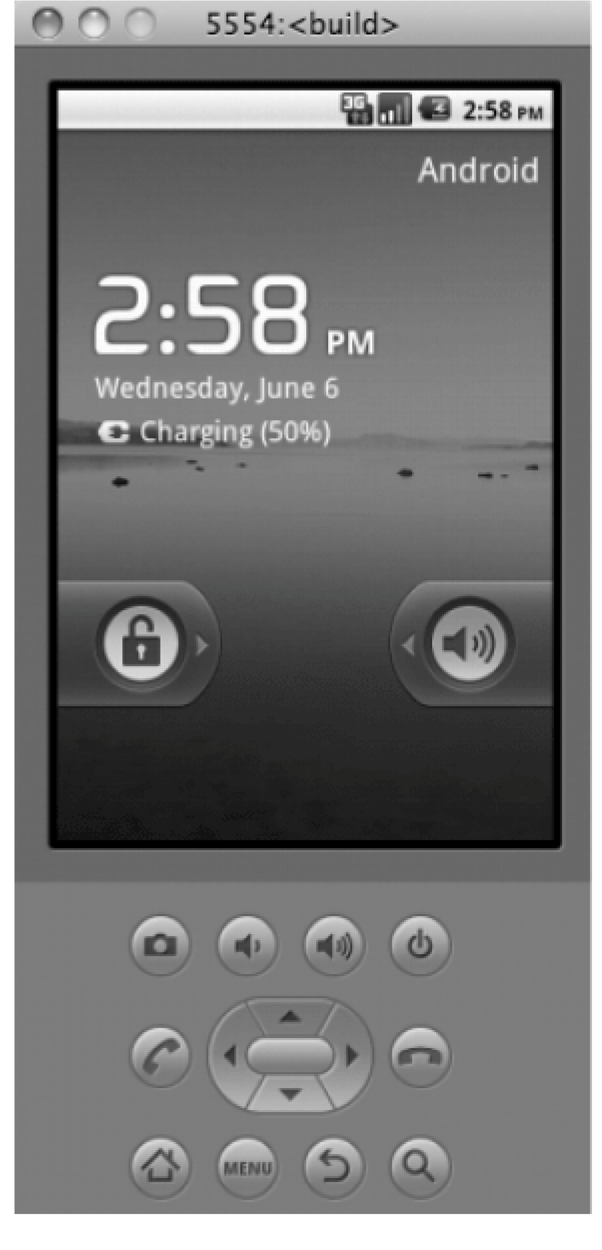


图 2.21 模拟器启动提示界面

稍后会弹出 Android 模拟器的运行界面,中央显示 ANDROID 字样。模拟器完全启动后,会显示 Android 锁定界面,向右滑动解锁图标解锁,如图 2.22 所示。





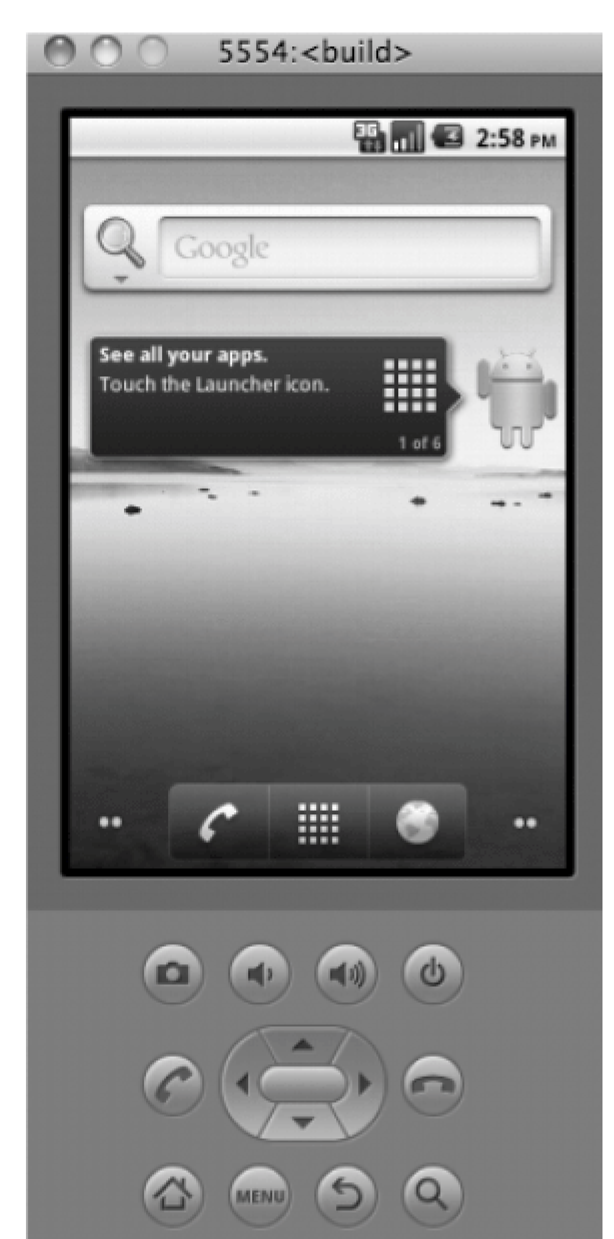


图 2.22　Android 模拟器

模拟器启动完成后，在模块编辑器的右上角单击 Connect to Device 按钮，从下拉菜单中选择刚启动的 Android 模拟器，笔者模拟器的编号为 emulator-5554，如图 2.23 所示。

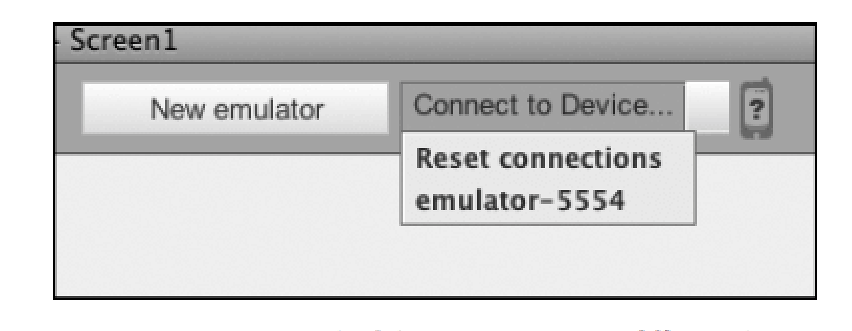


图 2.23　连接 Android 模拟器

Android 模拟器可以仿真手机的绝大部分硬件和软件功能，支持加载 SD 卡映像文件，更改模拟网络状态、延迟和速度，模拟电话呼叫和接收短信等，支持将屏幕当成触摸屏使用，可以使用鼠标单击屏幕模拟用户对 Android 设备的触摸操纵。在 Android 模拟器上有普通手机常见的各种按键，如音量键、挂断键、返回键和菜单键等。但 Android 模拟器仍不支持的功能包括：接听真实电话呼叫、USB 连接、摄像头捕获、连接状态检测、电池电量、AC 电源检测、SD 卡插拔检查和蓝牙设备等。

2.5.2　USB 连接手机

除了使用 Android 模拟器调试程序以外，用户也可以使用 Android 手机调试程序。在进行手机调试前，先要将 Android 手机与 App Inventor 连接在一起，这样在 App Inventor 中开发的程序才可以自动安装到手机里面。与手机连接的途径有两种，分别是通过 USB 数据线连接和通过 WiFi 网络连接。

使用 USB 数据线连接 Android 手机是一种比较简便的方法，但需要用户对手机进行一些设置。

首先，进入“设置”页面，单击“应用程序”，接着勾选应用程序内的“未知来源”选项，这样可以允许非谷歌应用程序商店中的软件在设备上运行。

在“设置”页面中进入“开发者选项”，勾选其中的“USB 调试”和“保持唤醒状态”这两个选项。勾选这两个选项可以保证在开发时能够进入调试状态，同时在充电模式下不进入休眠状态。

设置完成后用数据线将手机与计算机相连，然后进入 App Inventor 模块编辑器，单击右上角的 Connect to Device 下拉菜单，如图 2.24 所示。用户的 Android 设备型号被识别后出现在下拉选框内，选择该手机型号即可将手机与 App Inventor 相连。

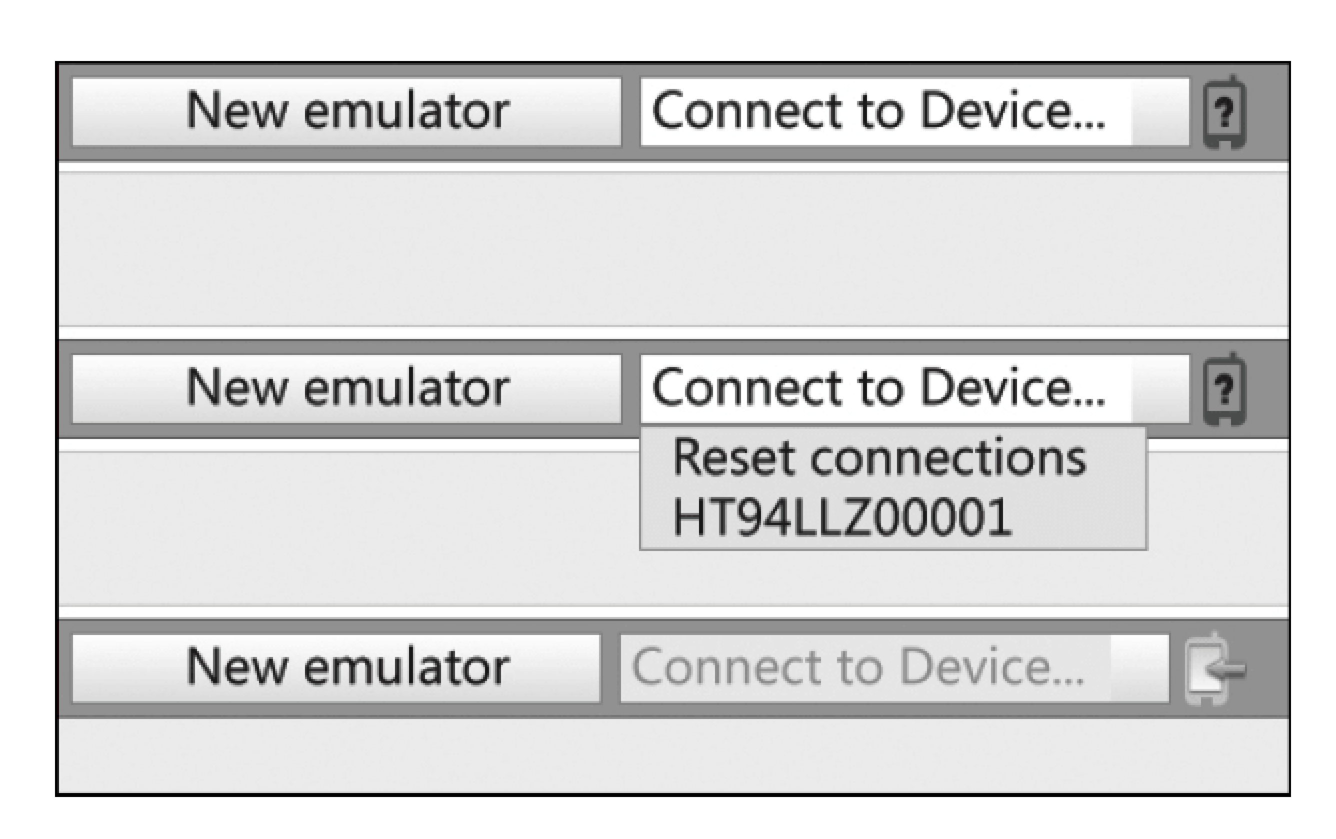


图 2.24　USB 连接手机

2.5.3　WiFi 连接手机

通过 WiFi 连接手机是 App Inventor 推荐的连接方式，用户只需要在手机上下载并安装 MIT ALCompanion 的软件，就可以让手机实时地从 App Inventor 中自动获取应用程序，简化调试过程。

在 Google Play 的 MIT ALCompanion 软件安装页面中，单击"安装"按钮，将该软件下载安装到 Android 手机上，如图 2.25 所示。MIT ALCompanion 的下载地址为：https://play.google.com/store/apps/details?id=edu.mit.appinventor.aicompanion2。

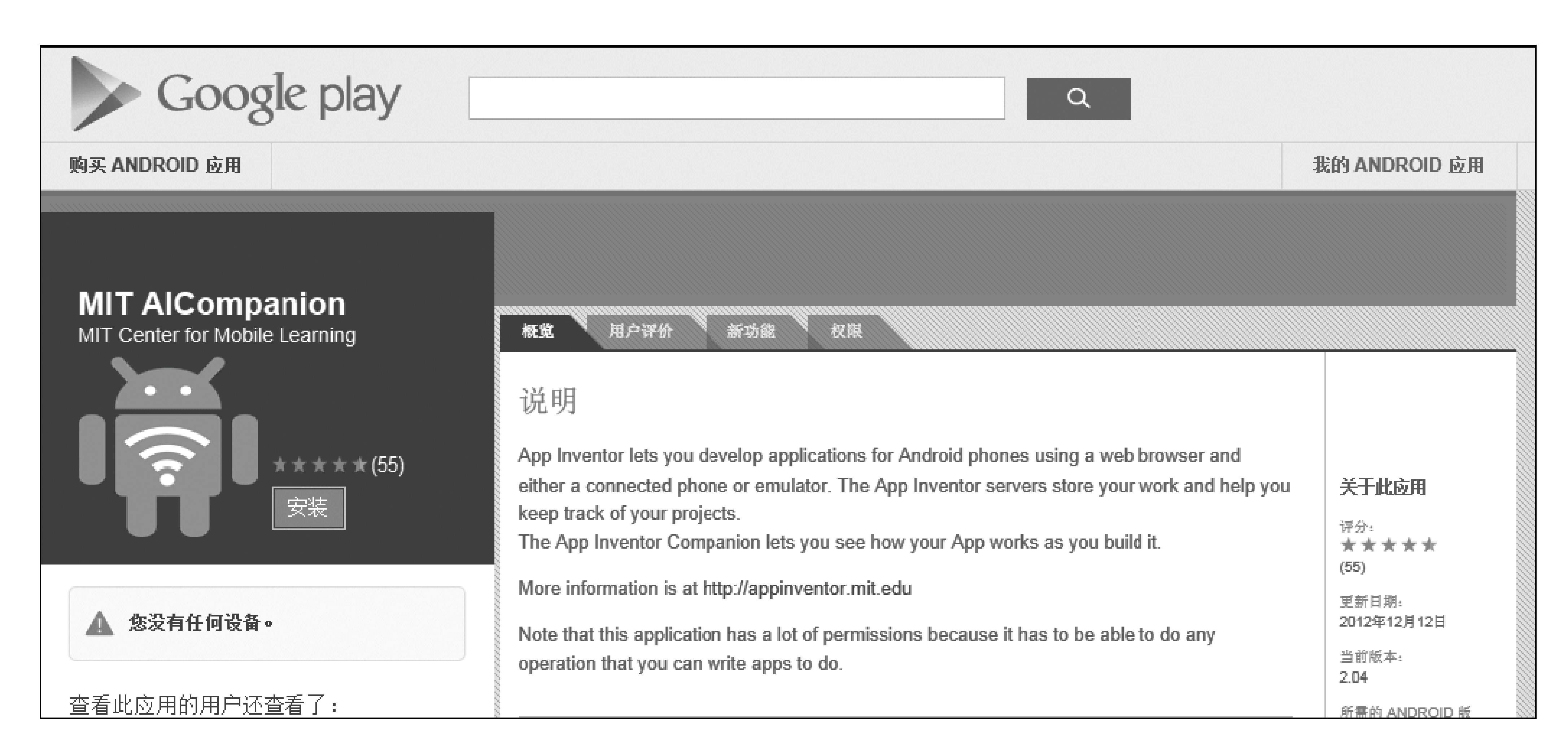
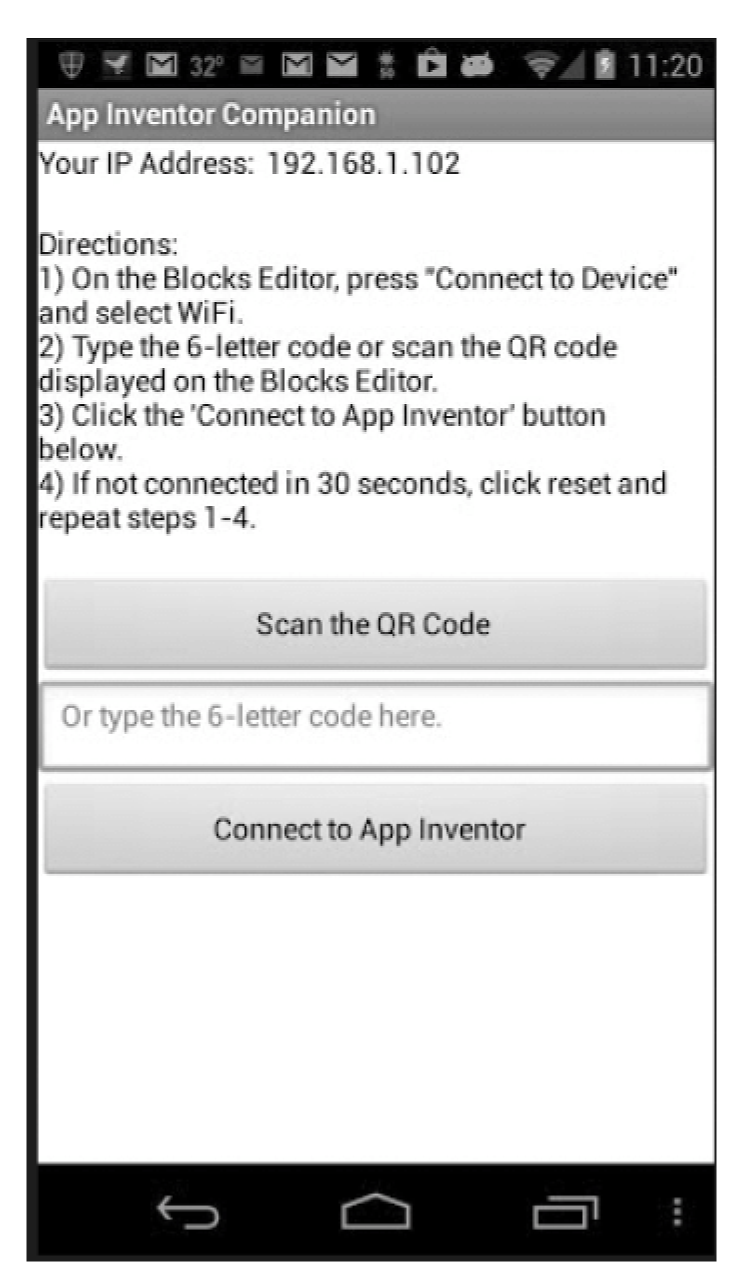


图 2.25　MIT ALCompanion 软件安装页面

完成 MIT ALCompanion 安装后，在手机上运行该软件，软件启动后的界面如图 2.26 所示。运行 MIT ALCompanion 前，需要开启手机的 WiFi 功能，因为 MIT ALCompanion 是通过无线局域网将数据从 App Inventor 传递到手机中。

启动手机上的 MIT ALCompanion 以后，在 App Inventor 的模块编辑器中选择 Connect to Device 下拉菜单中 WiFi 选项，如图 2.27 所示。



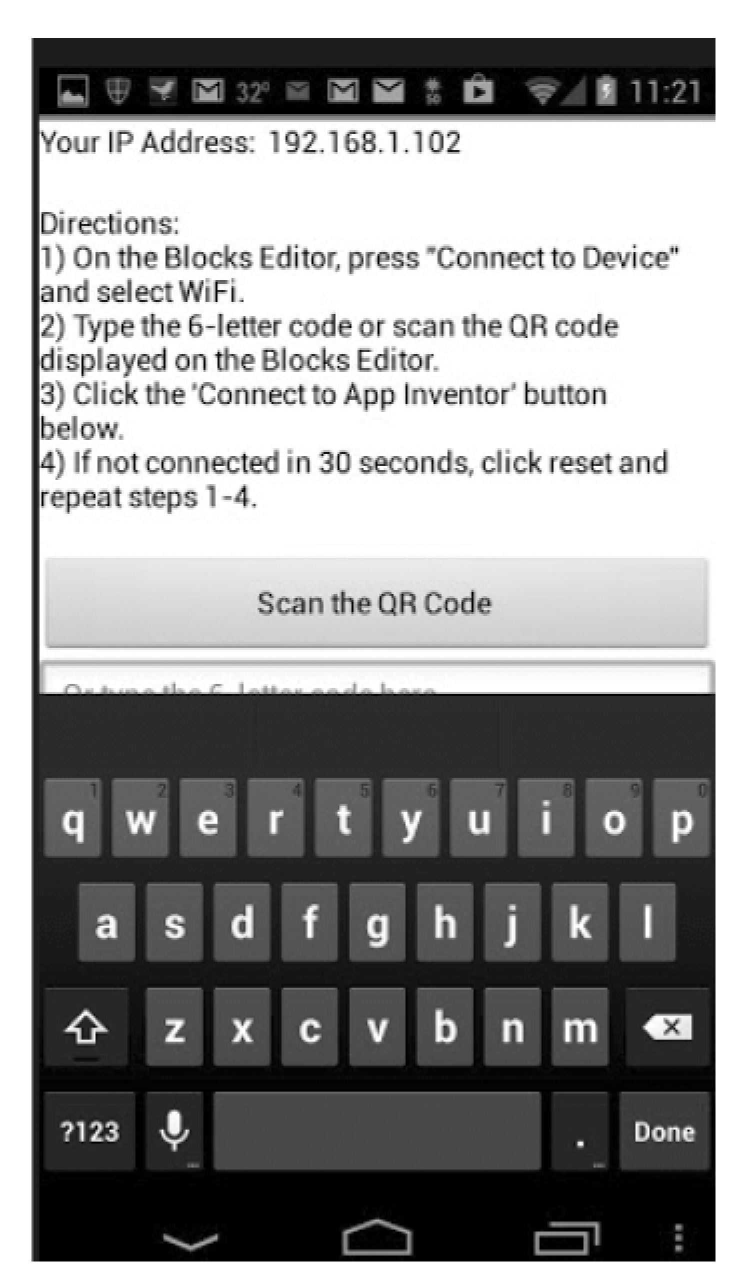


图 2.26 MIT ALCompanion

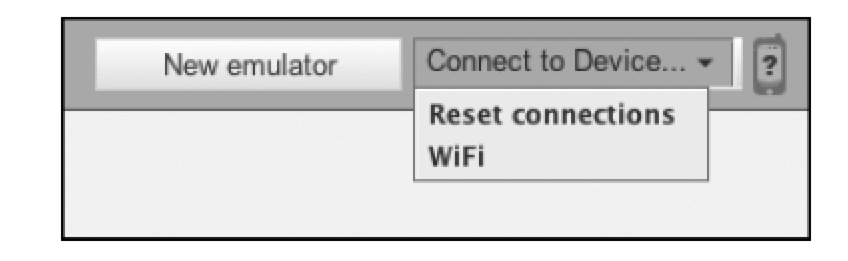


图 2.27 模块编辑器设备连接选项

计算机屏幕上会弹出一个对话框，上面提供了一个 6 位验证码和二维码，如图 2.28 所示。用户可以选择在手机的 MIT ALCompanion 中输入验证码与 App Inventor 取得连接，也可通过扫描二维码来获取连接。



图 2.28 设备连接的验证码和二维码

以上是利用 App Inventor 进行 Android 应用软件开发的前期准备工作，如果用户在操作中遇到任何问题，可以参考 MIT App Inventor 的 Troubleshooting Page，其网址为 http://appinventor.mit.edu/explore/content/troubleshooting.html/，希望以上介绍能够为 Android 应用开发学习者们带来帮助。

习　　题

1. 安装 App Inventor 开发环境，并记录安装和配置过程中所遇到的问题。
2. 尝试安装 App Inventor 程序，并分别使用模拟器和手机进行调试。
3. 分析使用 WiFi 连接手机调试程序的优缺点。

第 3 章



第一个 App Inventor 程序

本章主要介绍开发 App Inventor 应用程序的基础知识和基本方法。通过本章内容的学习，读者可以了解开发 App Inventor 应用程序的过程和方法，进一步掌握 App Inventor 程序的界面开发方法和逻辑部分开发方法。

本章学习目标：

- 了解 App Inventor 的程序开发流程；
- 掌握界面编辑器的使用方法；
- 掌握模块编辑器的使用方法。

3.1 创建新工程

本章将详细介绍如何开发第一个 App Inventor 程序 HelloAppInventor，并在第 2 章的基础上，讲解如何使用 App Inventor 建立新工程、使用界面设计器开发用户界面、使用模块编辑器开发程序逻辑，以及使用手机或模拟器进行程序调试。



图 3.1　HelloAppInventor 示例界面

HelloAppInventor 示例非常简单，界面如图 3.1 所示，用户在界面上单击“请按我”按钮，则会在按钮下方出现“Hello！ App Inventor”。

进入 App Inventor 后，在 My Projects 页面中单击 New 按钮，新建一个 App Inventor 工程。在弹出的 New App Inventor for Android Project 对话框中，在 Project name 文本框中填入新建 App Inventor 工程的名称，这里填入“HelloAppInventor”作为工程名称，然后单击 OK 按钮完成工程创建，如图 3.2 所示。

工程创建完成后，Projects 中会显示刚刚创建的工程，如图 3.3 所示。工程名称 HelloAppInventor 前面有一个复选框，作用是进行批量删除操作，如果选择这个复选框，Delete 按钮就会变为可单击的状态。

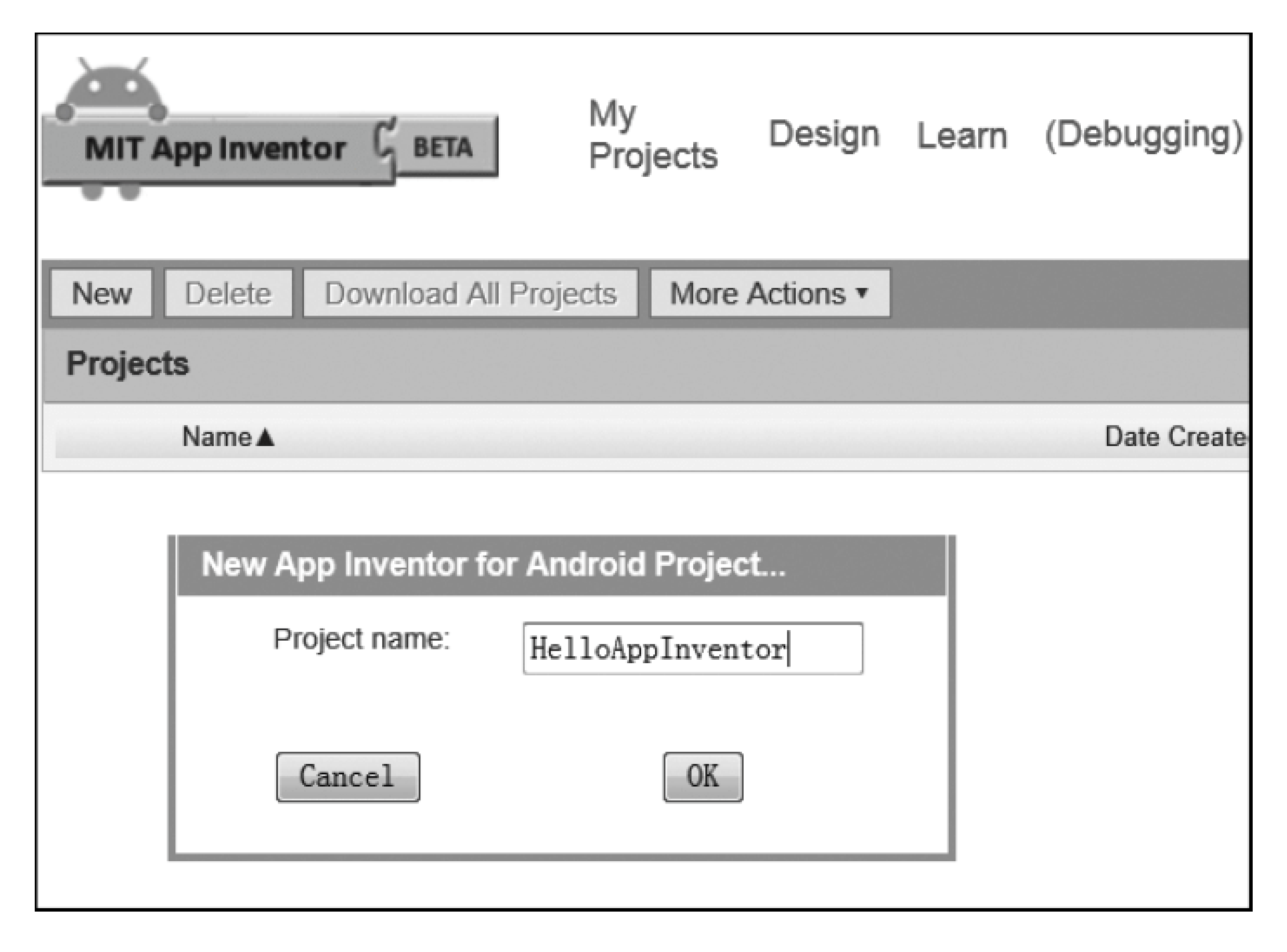


图 3.2　App Inventor 工程项目的创建



图 3.3　App Inventor 工程列表

3.2　界面设计

单击新创建的工程名称 HelloAppInventor，App Inventor 会打开界面设计器页面，如图 3.4 所示。这样，用户就可以开始进行 HelloAppInventor 的界面设计。

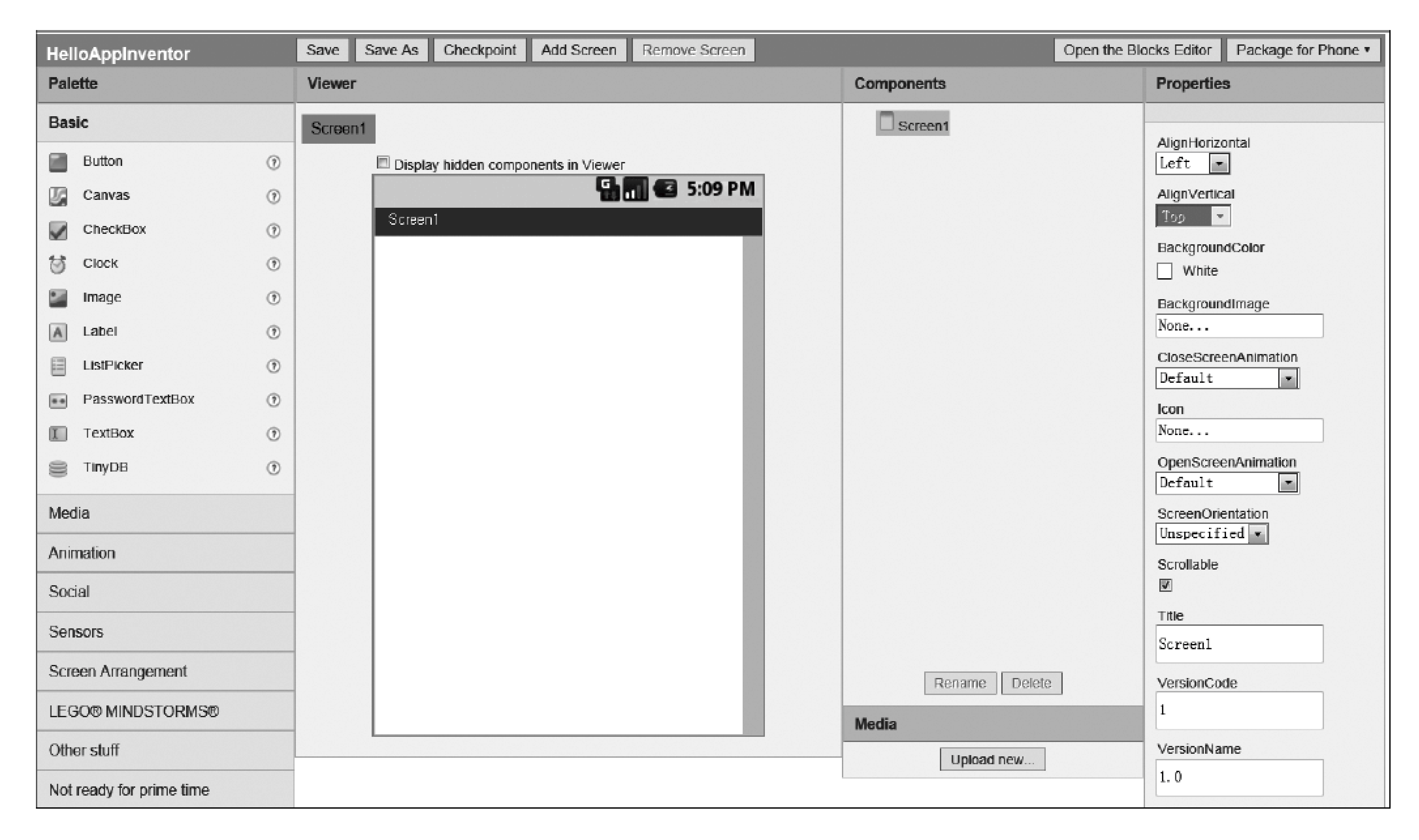


图 3.4　HelloAppInventor 示例的界面设计器

App Inventor 会自动创建一个屏幕页 Screen1(Android 系统中的 Activity),可以在模块区(Components)中看到屏幕页 Screen1。屏幕页是界面控件的承载体,供用户在其上面摆放各种界面控件。

在属性设置区(Properties)中,将 Screen1 屏幕页的标题(Title)属性从"Screen1"更改为"Hello AppInventor",设计区(Viewer)中 HelloAppInventor 示例的标题也会立即更改为"Hello AppInventor",如图 3.5 所示。

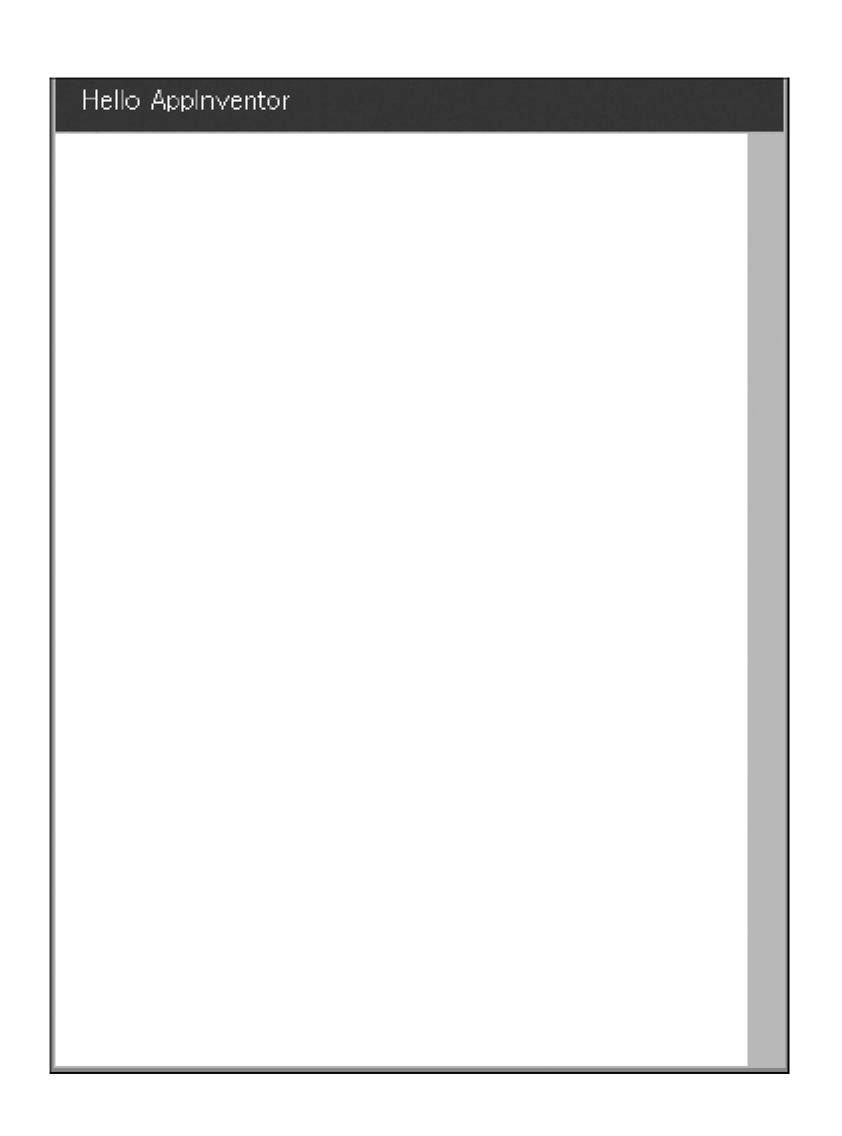


图 3.5 HelloAppInventor 示例的标题

接下来第二步是从左侧的控件库(Palette)的 Basic 区域中,将按钮(Button)控件拖曳到屏幕页 Screen1 上。屏幕页上立即会出现一个按钮,显示的内容是默认字符串"Text for Button1",如图 3.6 所示。

此时模块区中也会出现刚刚放置的按钮控件,如图 3.7 所示,默认的名称为"Button1"。这是一个较为简便的命名方式,用户使用的第一个按钮就命名为"Button1",第二个按钮就命名为"Button2",以此类推。如果界面上的控件较少,这样的命名方式也未尝不可,但如果界面较为复杂,建议给这些控件重新命名,在后面的逻辑设计中,可以方便地知道这些控件在界面上是做什么用的。例如,Button1 按钮,就可以重新命名为"ButtonClickMe"或是"button_click_me"等。

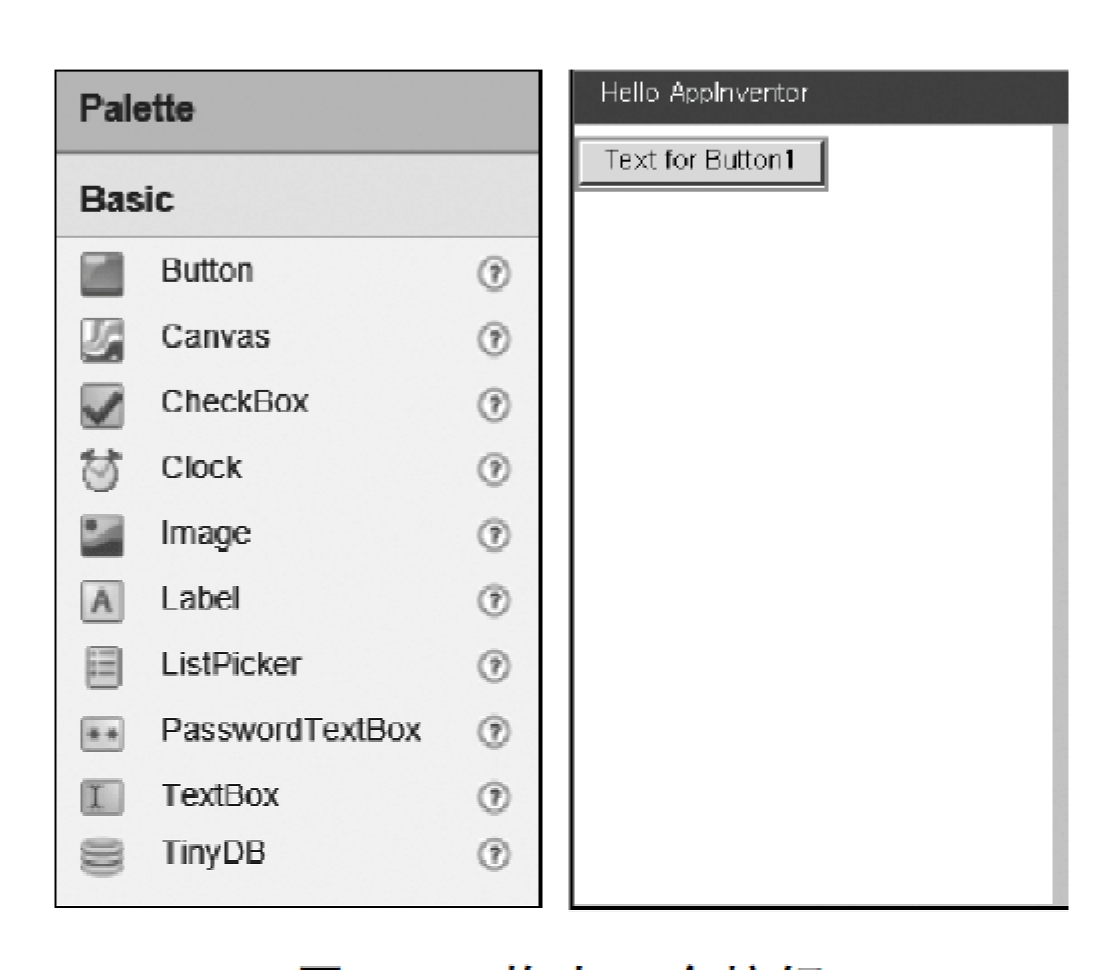


图 3.6 拖曳一个按钮

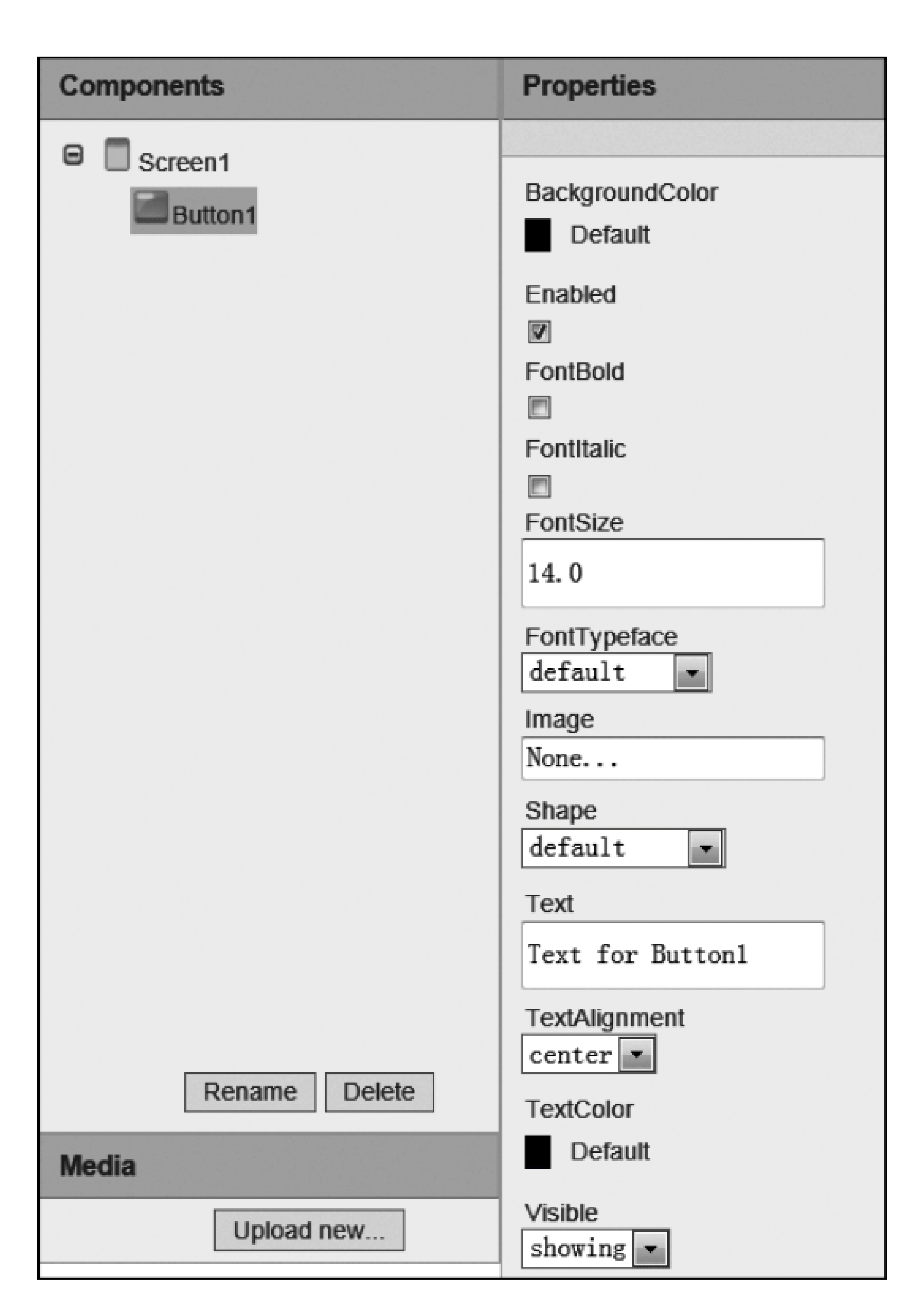


图 3.7 Button1 按钮模块和属性

下面修改属性设定区中 Button1 按钮的属性，所有的修改内容如表 3.1 所示。修改 Button1 按钮属性的目的是让按钮看起来更加醒目，且可以显示中文提示“请按我”，使用者就可以轻易地判读出这个控件是一个按钮。

表 3.1　Button1 按钮属性

属　性	默认值	修改值	属　性	默认值	修改值
BackgroundColor	Default	Green	Text	Text for Button1	请按我
FontSize	14.0	30	Width	Automatic	Fill parent

如图 3.8 所示，首先将按钮的背景颜色修改为绿色，修改方法是将 BackgroundColor 属性由默认的“Default”改为“Green”。修改背景颜色时会弹出颜色板，直接选择绿色就可以了。修改 Button1 按钮的字体大小是通过修改 FontSize 属性实现的，FontSize 属性的默认值是“14.0”，将其修改为“30”即可。Text 属性是控制按钮的显示内容，将其从默认的“Text for Button1”改为“请按我”。



图 3.8　Button1 按钮属性修改

Width 属性是控制按钮宽度的属性，默认值是 Automatic，表示按钮的宽度会自适应文字的长度。另一种设置方法是 Fill parent，这样的设置方法，按钮的宽度会填充满整个父控件，Button1 就采用这种设置方法。最后一种设置方法是将宽度设置为固定的值，这里用像素作为单位。

Button1 控件的属性设置完成后，设计区中生成的 HelloAppInventor 界面如图 3.9 所示。因为按钮宽度属性设置为 Fill parent，按钮的宽度达到了最大值。

第三步是从控件库的 Basic 组件区中，将一个标签(Label)控件拖曳到屏幕页中，标签控件一般用来显示文字。标签控件被拖曳到屏幕页后，会自动出现在前一个控件的下方。在 HelloAppInventor 示例中，标签控件会出现在 Button1 按钮下方，显示的文字为 Text for Label1，如图 3.10 所示。



图 3.9　设计区中的 HelloAppInventor 界面



图 3.10　添加标签后的界面

因为这是界面中的第一个标签，因此标签的名称为 Label1。Label1 标签的默认属性如图 3.11 所示。

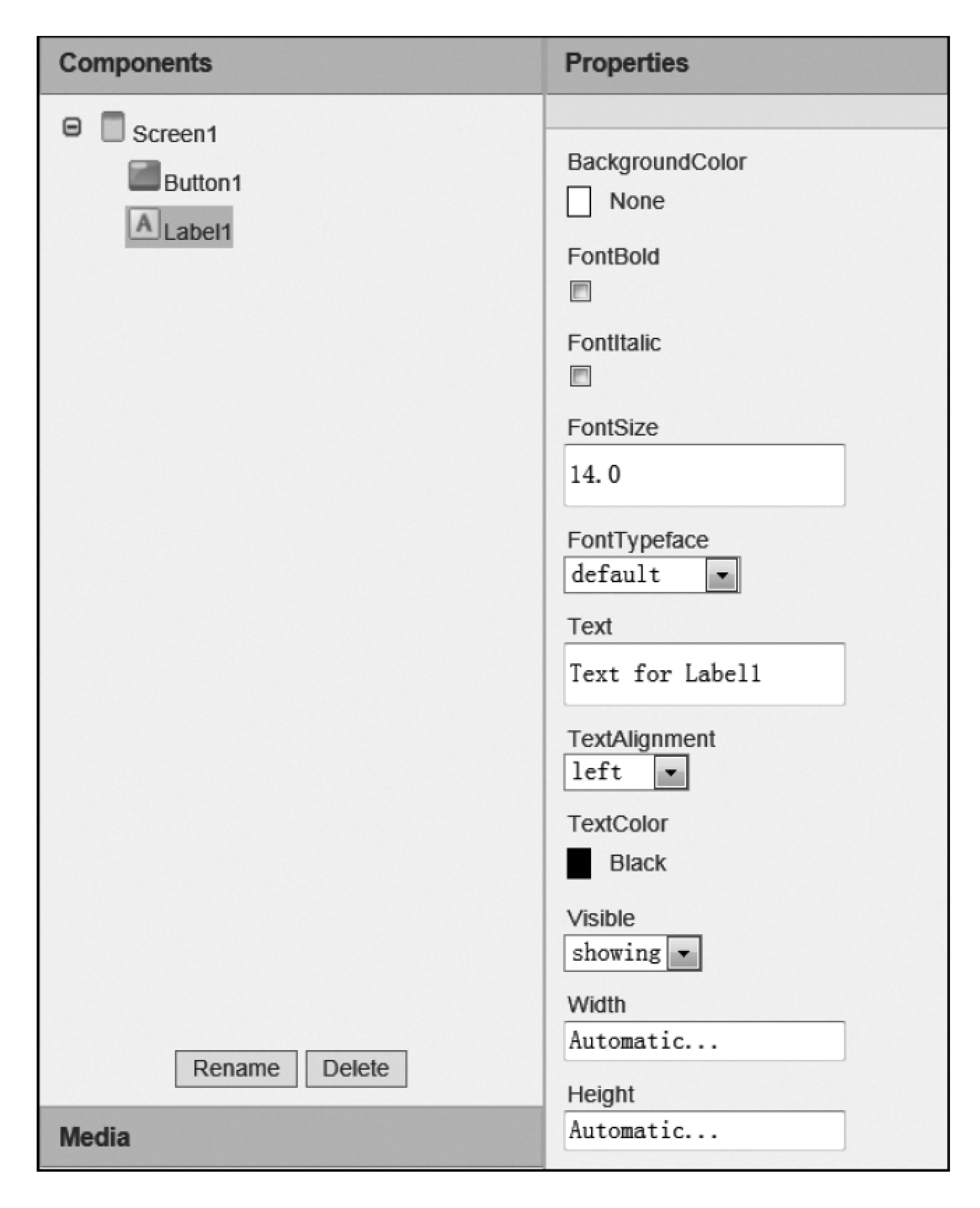


图 3.11　Label1 的属性

下面需要修改 Label1 标签的字号、显示文字和宽度，按照表 3.2 修改 Label1 标签的属性。

在 Label1 标签属性修改完毕后，Label1 标签因为 Text 属性为空，因此在界面上 Label1 标签已经不可见，但它还是确实存在的，如图 3.12 所示。



图 3.12　Label1 标签属性修改后的界面

表 3.2　Label1 属性

属性	默认值	修改值
FontSize	14.0	30
Text	Text for Label1	
Width	Automatic	Fill parent

至此，HelloAppInventor 示例的界面设计部分已经完成了，单击 Save 按钮保存界面设计。最后，单击 Open the Blocks Editor 按钮，打开模块编辑器，准备进行程序逻辑的设计。

3.3　逻辑模块开发

新工程打开的模块编辑器(Blocks Editor)如图 3.13 所示。模块编辑器的左上方有工程名称和屏幕页名称"HelloAppInventor-Screen1"，表示这个模块编辑器是 HelloAppInventor 示例的 Screen1 屏幕页的模块编辑器。

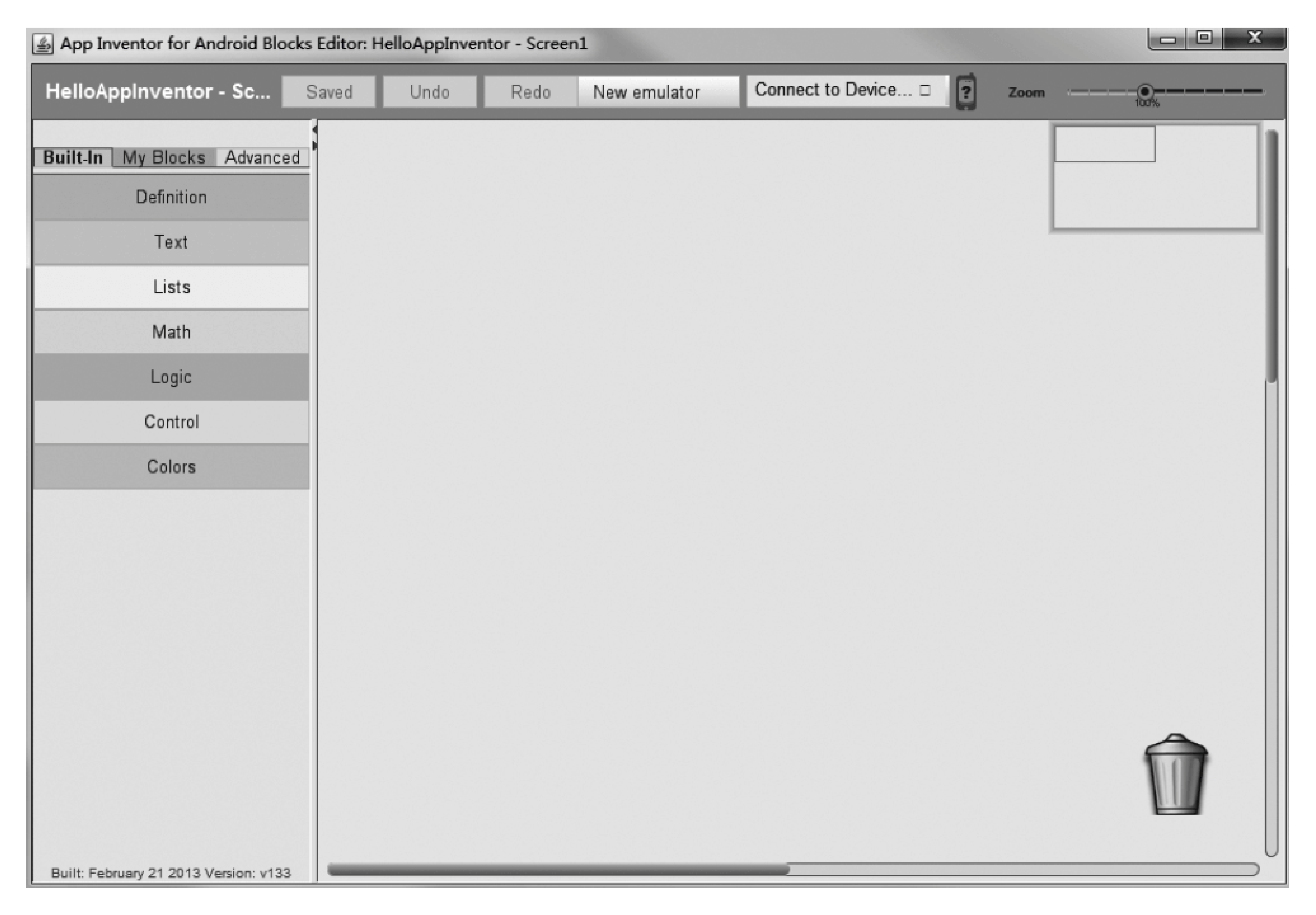


图 3.13　模块编辑器

在HelloAppInventor示例中，要实现的逻辑可以这么描述：在单击“请按我”按钮后，标签显示文字“Hello！ App Inventor”。

上面的这段逻辑描述中，可以找到三个关键的元素：“请按我”按钮、标签和文字“Hello！ App Inventor”；两个关键动作：“单击”和“显示”。“单击”动作对应“请按我”按钮控件，而“显示”动作对应标签控件。

如果这样分析，只要在模块编辑器中找到：按钮的单击、显示标签内容和内容的承载体，就可以完成上述逻辑。而事实上，这样的模块在模块编辑器中可以很容易找到，如图3.14所示。



图3.14 HelloAppInventor示例所需的模块

下面来说明如何在模块编辑器的模块编辑库中，找到程序逻辑设计所需要的模块。在模块编辑库中分为Built-In、My Blocks和Advanced三个子项，如图3.15所示。

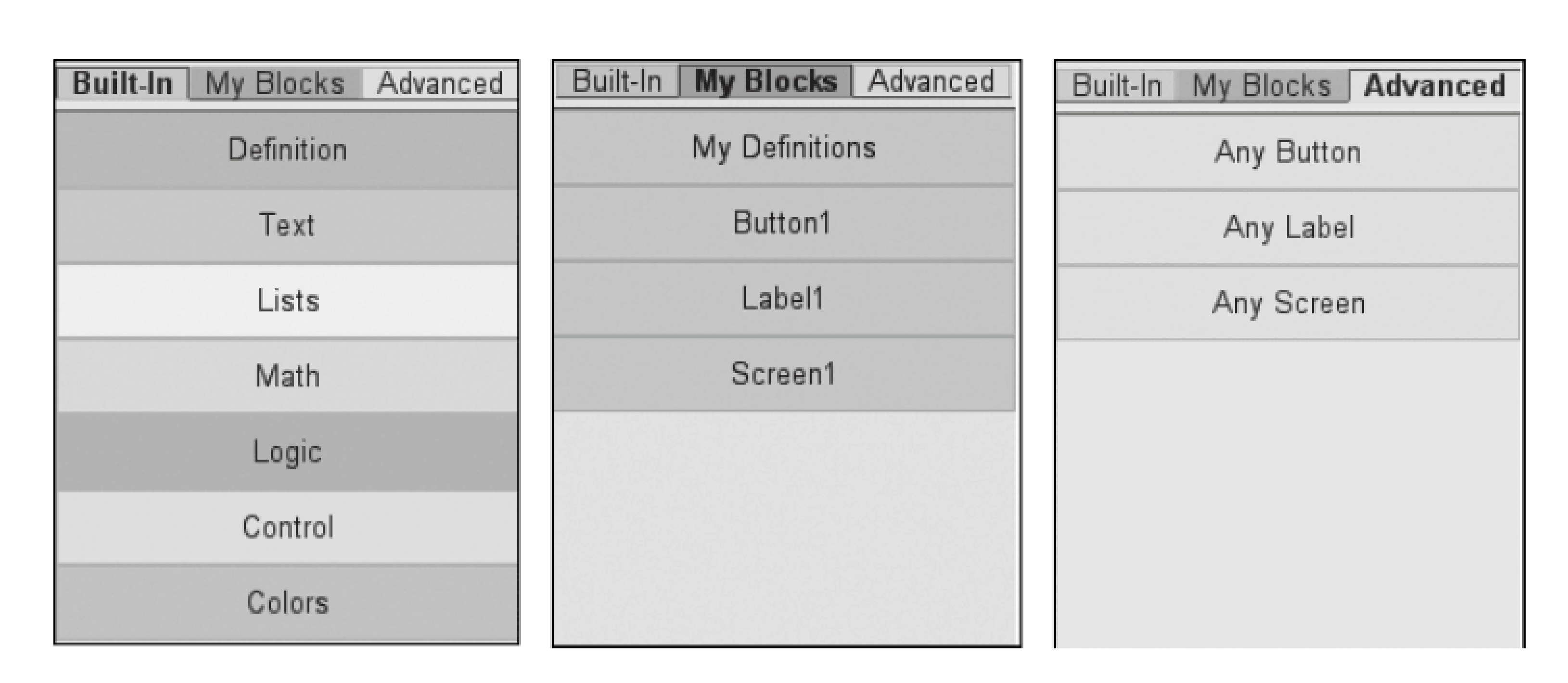


图3.15 模块编辑库

Built-In是内建模块，经常使用到的模块被保存在这里，包括字符串模块、列表模块、逻辑模块、控制模块和颜色模块，以及用来建立可复用程序的函数模块。

My Blocks是用户界面中控件的事件模块和方法模块，这里的模块类型和数量会根据用户界面中所包含的控件数量变化。

Advanced是高级模块，基本上是对所有同类型控件的操控，例如全部按钮、全部标签或是全部屏幕页。

回想一下刚才对逻辑的描述：在“请按我”按钮被单击后，标签显示文字“Hello！App Inventor”。“请按我”按钮的名称是Button1，这样首先找到第一个模块：“Button1的点击”。

第一个模块可以按照如下方式找到：My Blocks→Button1→Button1.Click，如图3.16所示。用户只需要单击My Blocks，然后单击Button1，最后将Button1.Click拖曳到右侧的设计区中即可。

准确地讲，Button1.Click模块是按钮的单击事件，这个模块会在按钮被单击的时候调用。除了按钮的单击事件以外，在图3.16中，还可以找到按钮的获取焦点事件、长时间单击事件和失去焦点事件。

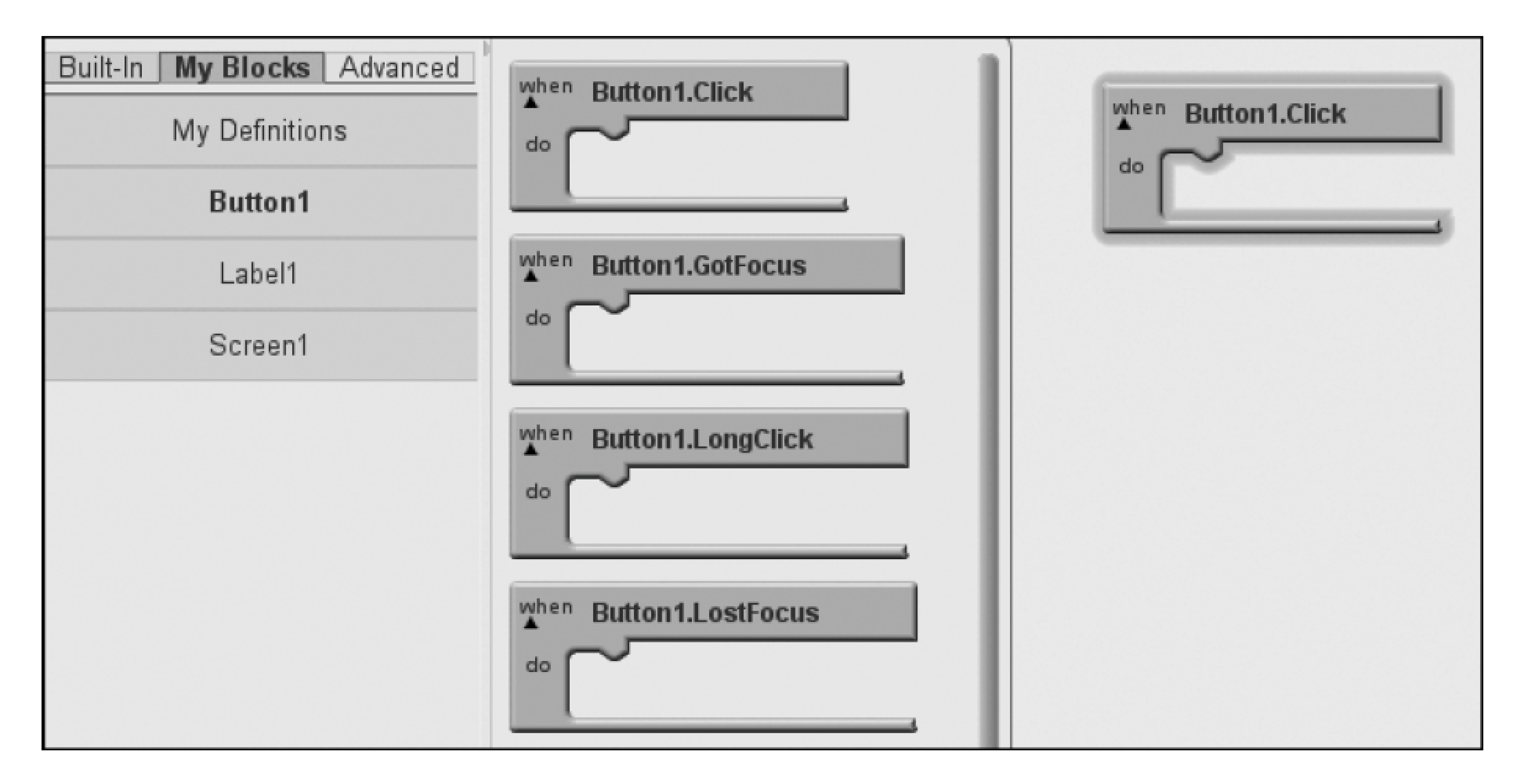


图 3.16 获取 Button1.Click 模块

第二个模块可以按照如下方式找到：My Blocks→Label1→Label1.Text，如图 3.17 所示，获取方法与获取 Button1.Click 模块的方法相似。Label1.Text 模块是一个方法，表示将 Label1 的显示内容进行更改。

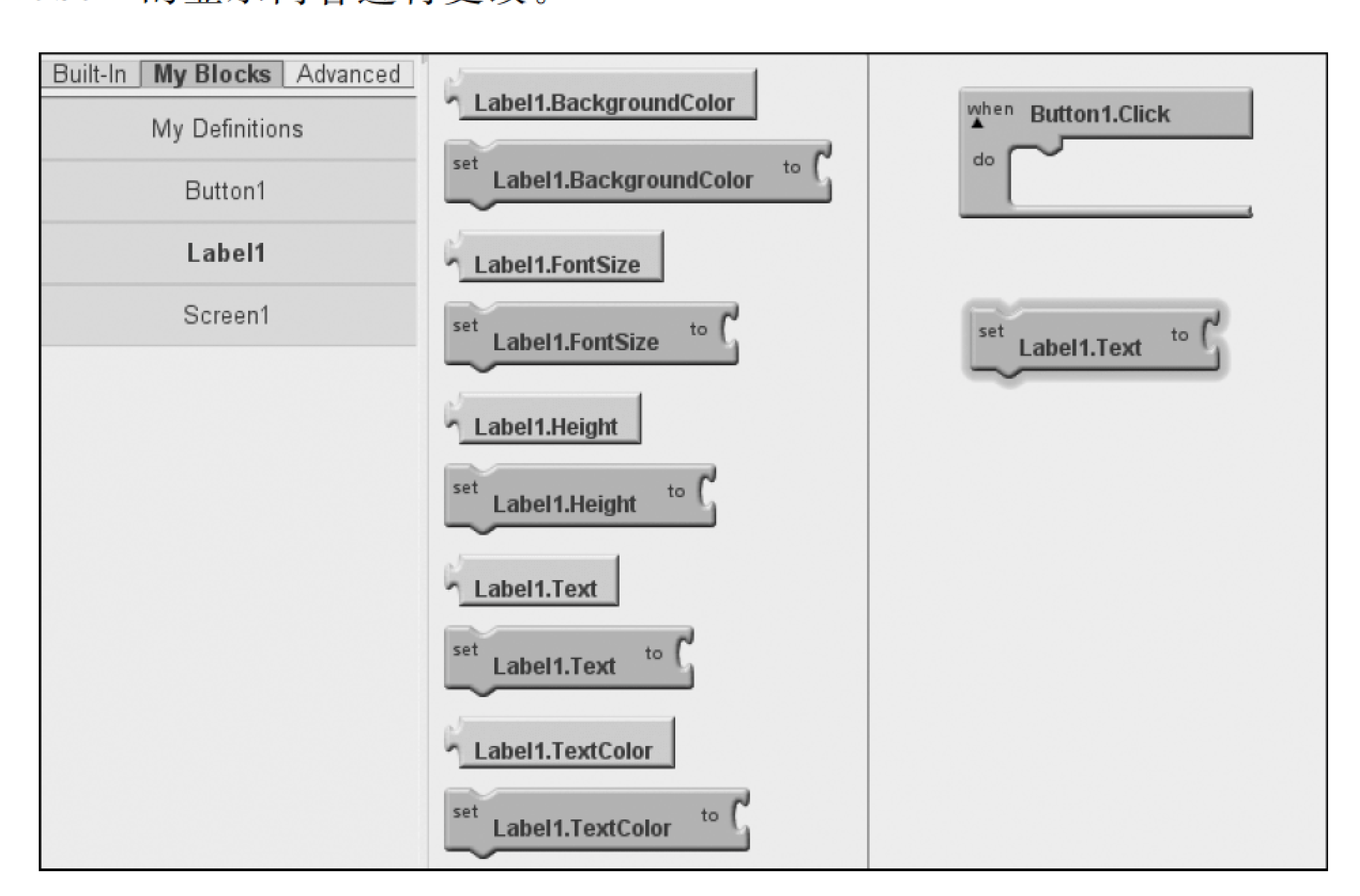


图 3.17 获取 Label1.Text 模块

第三个模块是 text 模块，用来表示需要显示的字符串，获取该模块的方法如下：Built-in→Text→text，如图 3.18 所示。

text 模块的默认显示内容为“text”，将其内容修改为“Hello! App Inventor”。用户只要单击 text 模块的文字部分，就可以进行修改，如图 3.19 所示。

在获取到所有需要的模块后，要将这些模块按照逻辑关系组装在一起，其过程就像拼图游戏，拼装过程如图 3.20 所示。

在拼装过程中不难发现，Button1.Click 模块和 Label1.Text 模块的卡槽是互相吻合的，这样的模块是可以拼装在一起的，同样 Label1.Text 模块和 text 模块的卡槽也是互

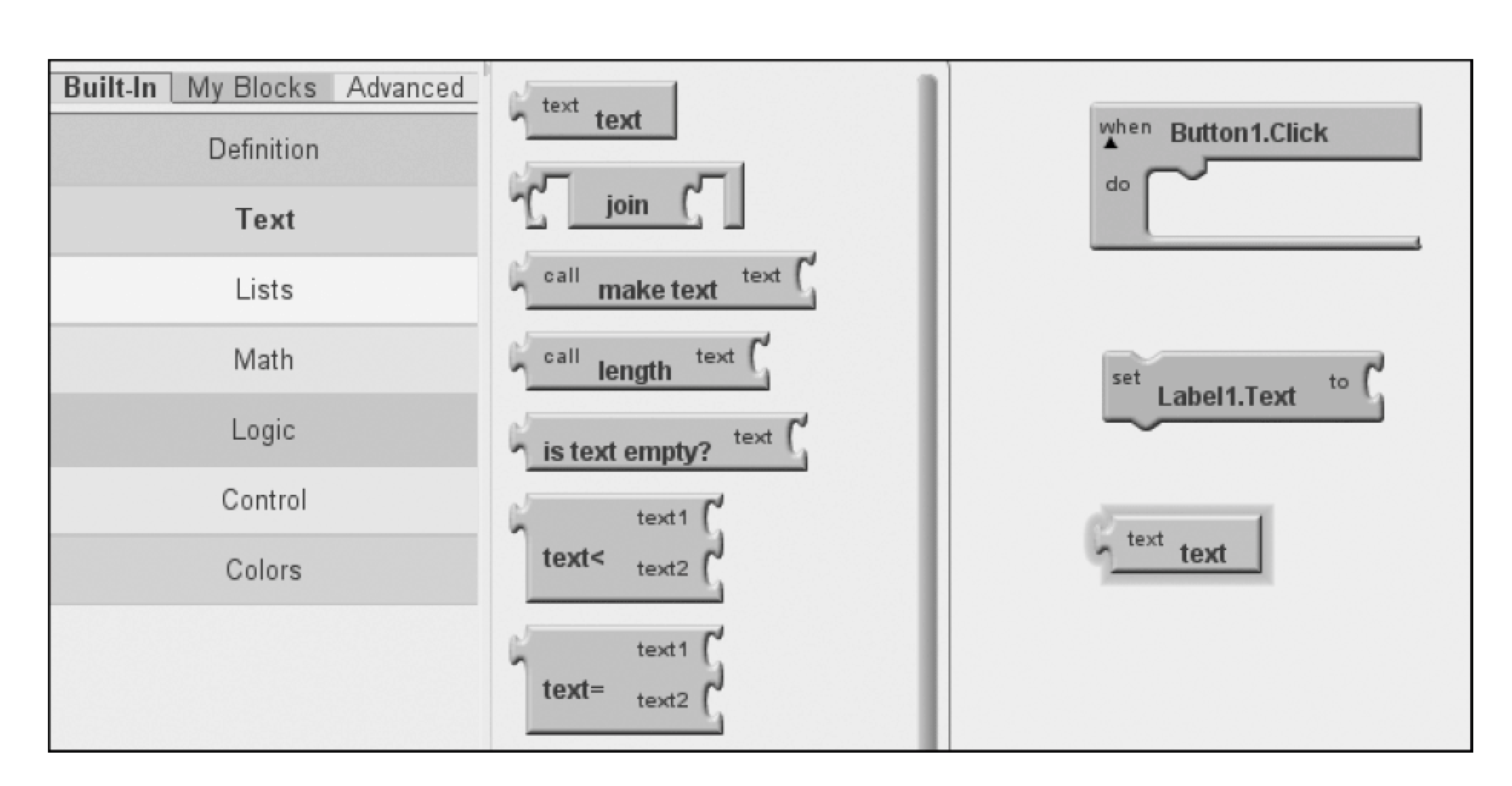


图 3.18 获取 text 模块

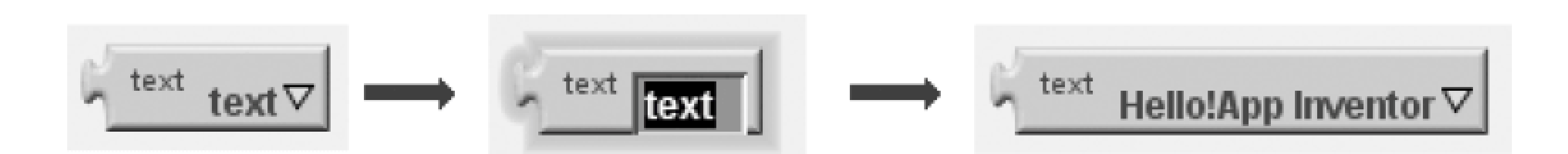


图 3.19 修改 text 模块参数

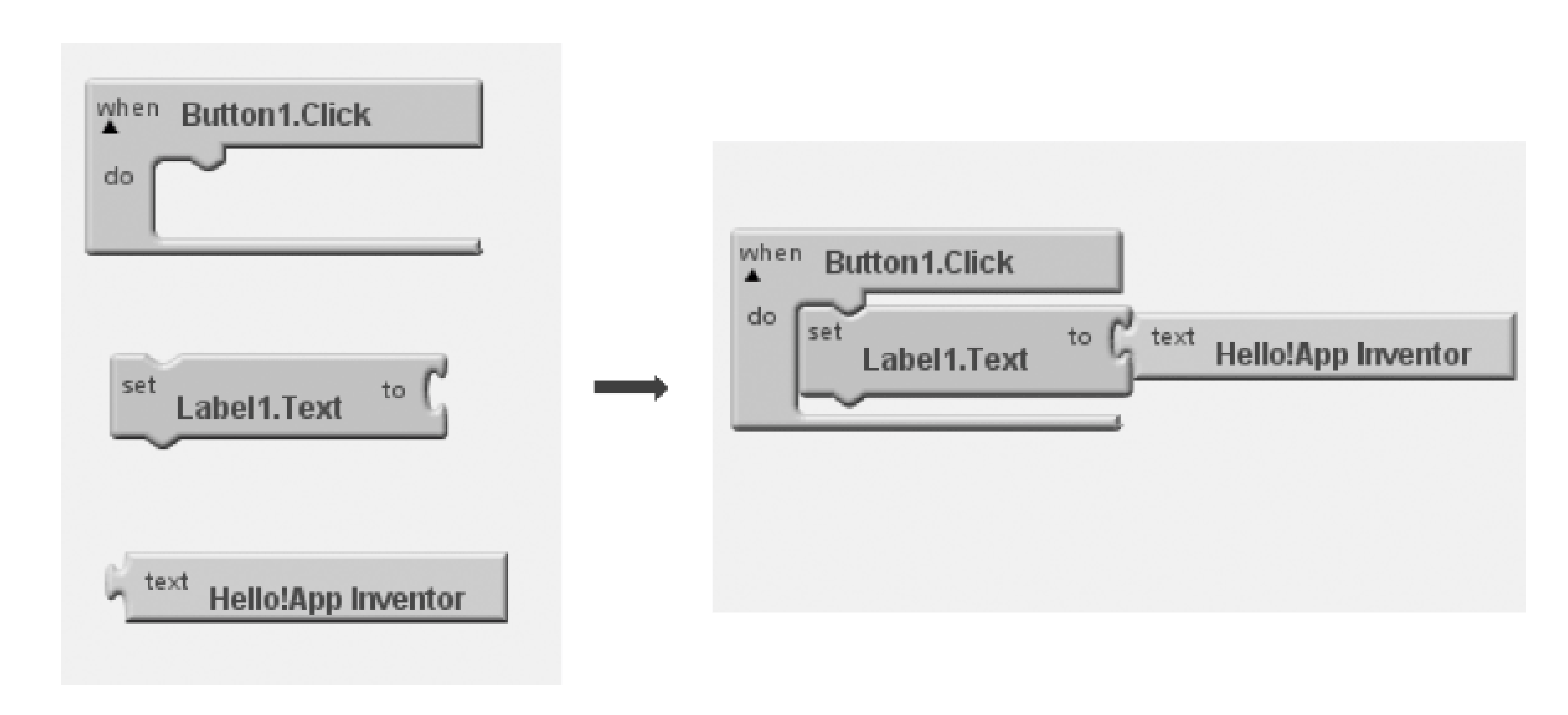


图 3.20 模块拼装

相吻合的。相反,Button1.Click 模块和 text 模块的卡槽并不匹配,则这两个模块无法拼装在一起。

App Inventor 中,如果两个模块成功地拼装在一起,将会发出清脆的“咔”的声音,而且两个模块的边缘是完全咬合在一起的。反之,如果两个模块无法拼装,则不会有任何反应。

拼装完成后,应单击屏幕左上角的 Saved 按钮,保存已经编辑好的程序逻辑。

3.4 程序调试

App Inventor 支持模拟器调试和手机设备调试,如果条件允许,尽量选择手机设备进行调试。因为模拟器在响应速度、显示效果和对硬件支持方面存在一定的不足,且使用手机调试完毕后,开发者可以将程序安装在手机中,与朋友和家人分享。

如何使用模拟器和手机进行调试的相关内容，可以参考第 2.5 节。笔者采用 USB 连接手机的方式进行调试，这种方式无须无线网络，而且不需要输入验证码，似乎更加易用一些。

USB 数据线将手机连接到所使用的计算机上后，单击模块编辑器右上方的 Connect to Device 按钮，在下拉菜单中选择手机的串号，本实例中手机设备的串号显示为“?”，如图 3.21 所示。

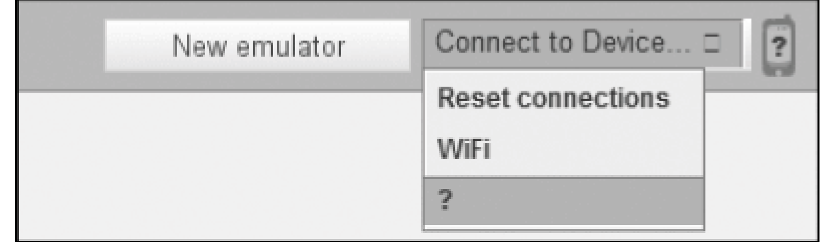


图 3.21　通过 USB 连接手机设备

App Inventor 通过 USB 与手机设备，当 Connect to Device 按钮右侧的电话图标由闪动的黄色变为静态的绿色时，表示程序与手机设备完成连接，此时手机设备屏幕显示手机“正在等待 App Inventor 命令，应用程序即将显示”，如图 3.22 所示。

片刻后手机屏幕显示 HelloAppInventor 示例的界面，单击“请按我”按钮后，将在按钮下方显示“Hello！ App Inventor”字样，如图 3.23 所示。

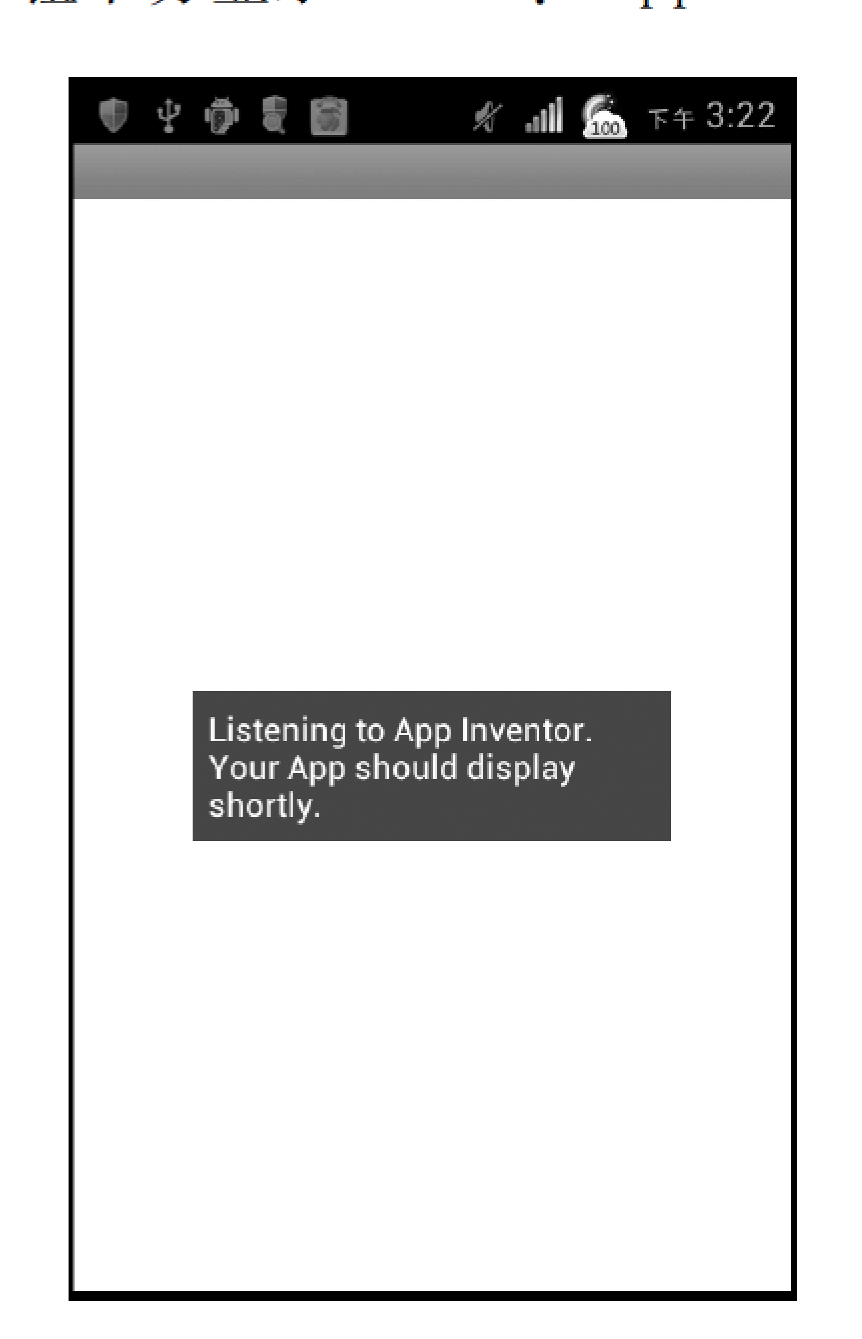


图 3.22　程序连接完成等候页面

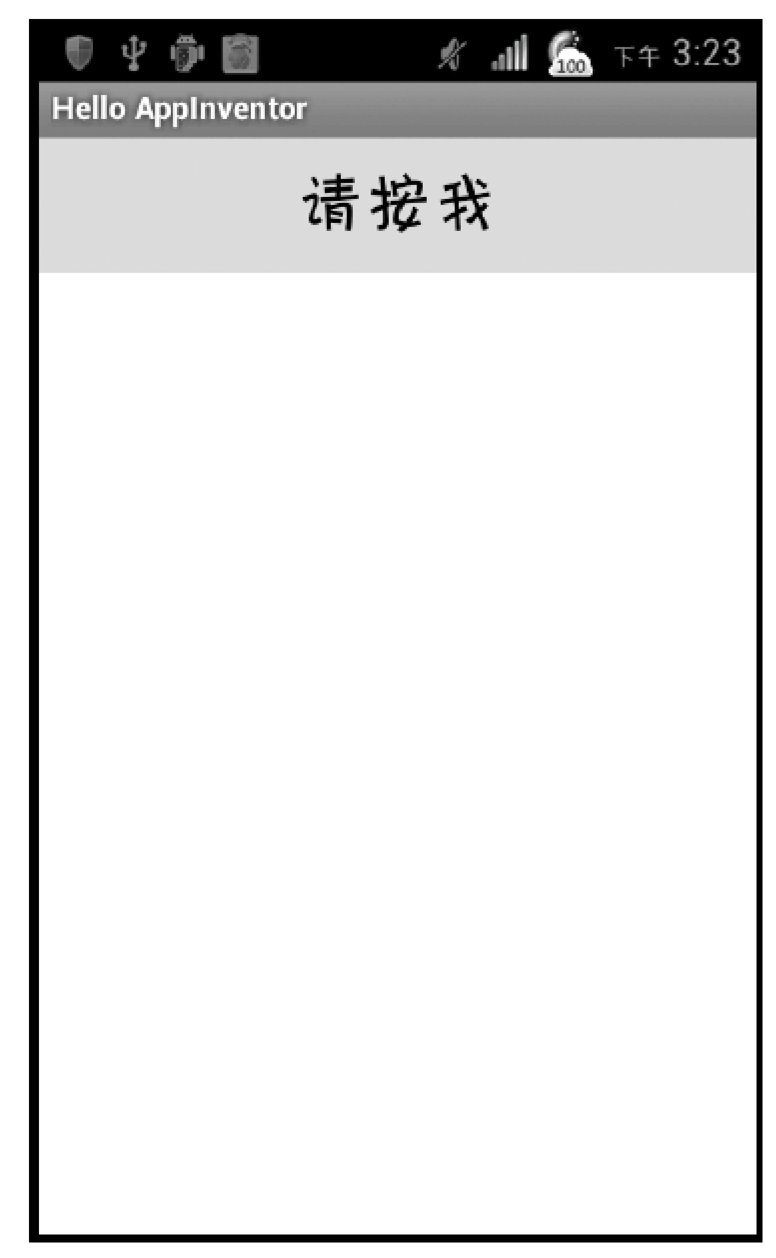




图 3.23　HelloAppInventor 示例的运行结果

习　　题

1. 使用 App Inventor 开发应用程序，界面编辑器和模块编辑器的功能各是什么？
2. 分析哪种程序调试方法更加适合你？为什么？
3. 尝试将 HelloAppInventor 示例的源代码、apk 文件下载到本地电脑，并想办法将 apk 文件安装在手机上。

第 4 章



程序设计基础

在程序设计过程中，不可避免地会使用到条件判断、循环、函数、列表和事件处理等控制执行顺序的模块。通过本章内容的学习，读者可以基本掌握这些模块的使用方法和场景。

本章学习目标：

- 掌握条件判断模块的使用方法；
- 了解布尔表达式；
- 掌握列表的使用方法，以及列表的分类和区别；
- 掌握循环模块的使用方法；
- 掌握函数模块的使用方法，以及函数的参数和返回值；
- 了解控件的事件、属性和方法的区别。

4.1 程序设计的基本结构

在 App Inventor 的开发环境中，程序设计的逻辑部分主要体现在模块编辑器(Blocks Editor)的设计环节中，是将选定的模块通过判断、循环、函数、列表和事件处理等程序结构拼接成完整的、正确的可执行程序。

在模块编辑器中，判断、循环、函数、列表和事件处理等程序结构同样是以模块的形式给出的，用户选择所需的结构模块，并将这些模块与组件模块相互拼接，最终实现完整的程序逻辑。其中，判断、循环、函数、列表和事件处理等类型的程序结构模块，在模块编辑器中模块集的 Built-In 中，并按照不同的分类将其分配在 Definition、Text、Lists、Math、Logic 和 Control 等分支中。

4.2 条件判断

在生活中经常会遇到下面这些说法：“如果明天天气好，我们就去郊游；如果天气不好，就不去了”，“如果成绩能到 90 分，就会奖励你；当然如果成绩还没有达到 60 分，就会批评你”。这些说法中的“如果……”就是条件判断。“如果”后面描述的事情称为“条

件”,所以这样的句子经常的形式为“如果+条件,条件成立时可以做的事情”。

在程序设计中,为了处理各种可能发生的情况,App Inventor 提供了两种条件判断模块:if 模块和 ifelse 模块,如图 4.1 所示。

这两种条件判断模块虽然起到条件分支的作用,但二者的具体使用情景却不相同,后面的内容将对两个条件判断模块分别进行介绍。

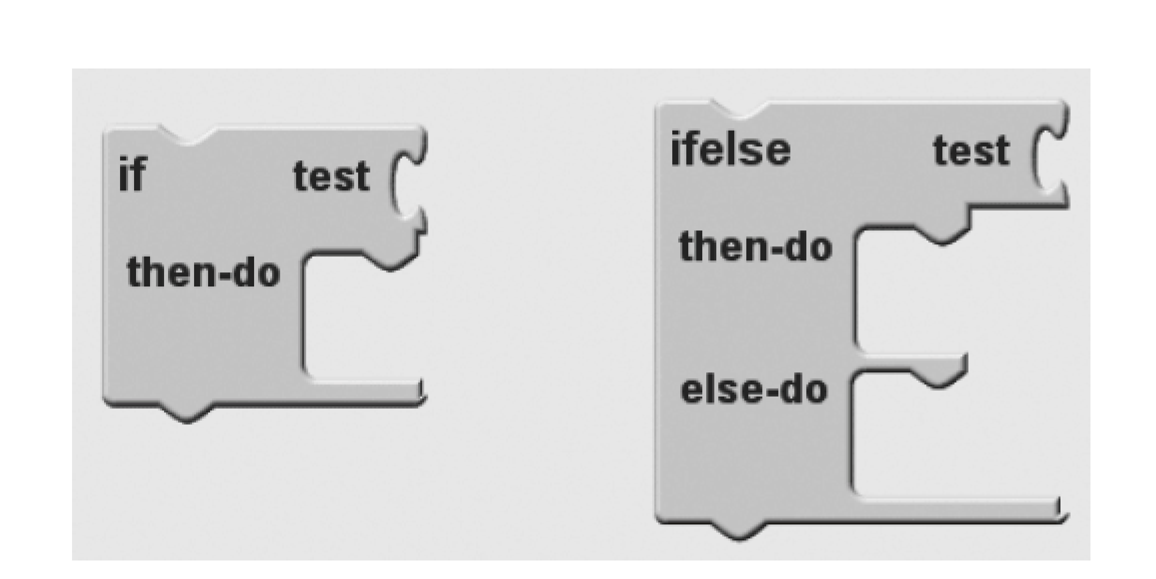


图 4.1 条件判断模块

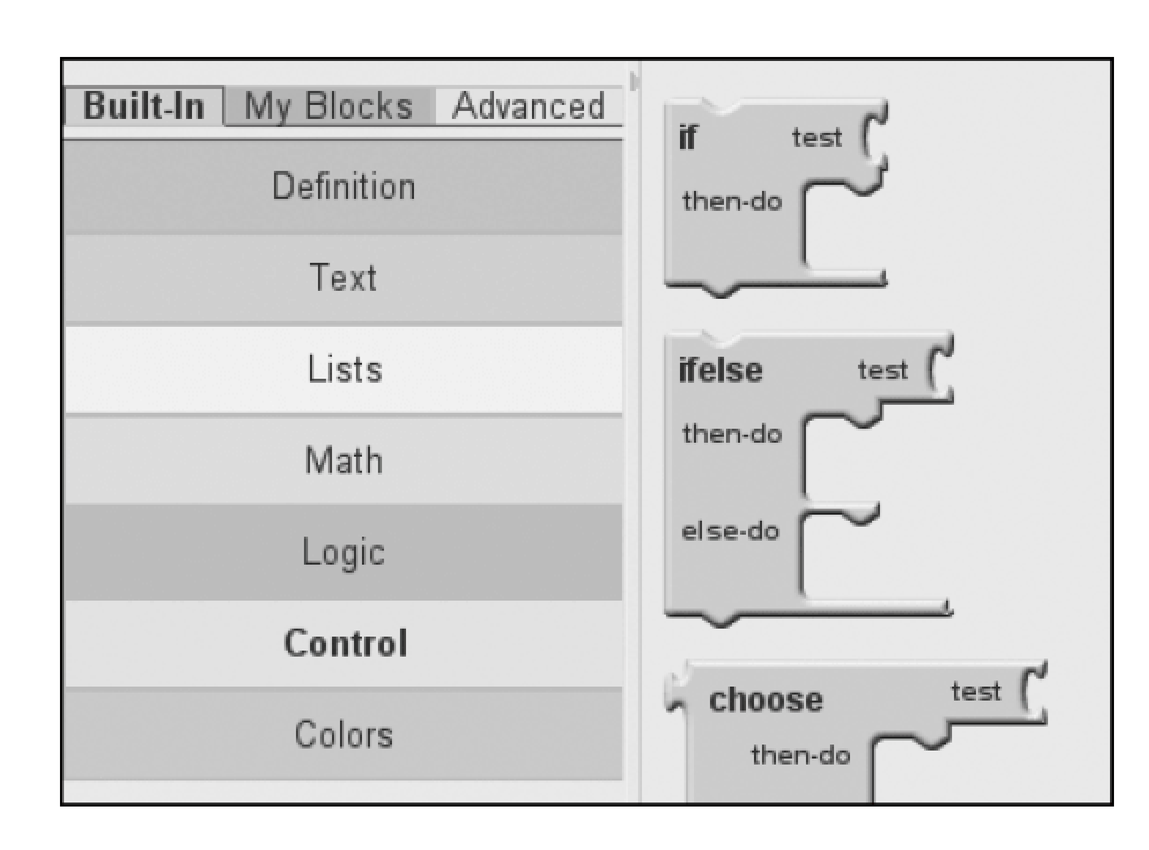


图 4.2 模块编辑器中的 if 模块

4.2.1 if 模块

在模块编辑器中,if 模块的位置在 Built-In→Control→if,如图 4.2 所示。

if 模块结构和执行流程如图 4.3 所示,其中槽 test 用来拼接条件,槽 then-do 用来拼接需要执行的动作。

if 模块的执行流程是先判断槽 test 的条件是否为 true,也就是说条件是否成立,如果条件成立,则执行槽 then-do 中的其他模块;如果条件为 false(条件不成立),则不执行槽 then-do 中的其他模块。下面给出一个小示例说明如何使用 if 模块。

在这个示例中,程序结构中只包含一个 if 模块,如图 4.4 所示。条件是“成绩小于等于 60 分”,条件为“score<=60”。score 是一个全局变量,在这里代表成绩分数。如果 score 小于等于 60,返回 true,if 模块将执行 then-do 中的内容,将 Label1 标签的显示内容修改为“failed”。反之,如果条件不成立,则返回 false,if 模块跳过 then-do 中的内容。

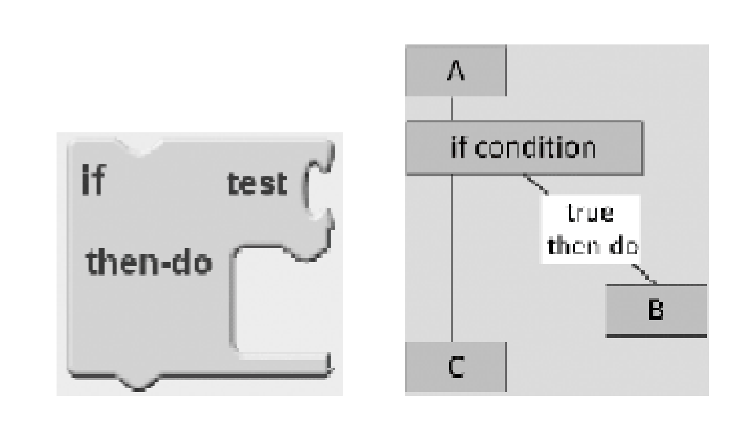


图 4.3 if 模块与执行流程

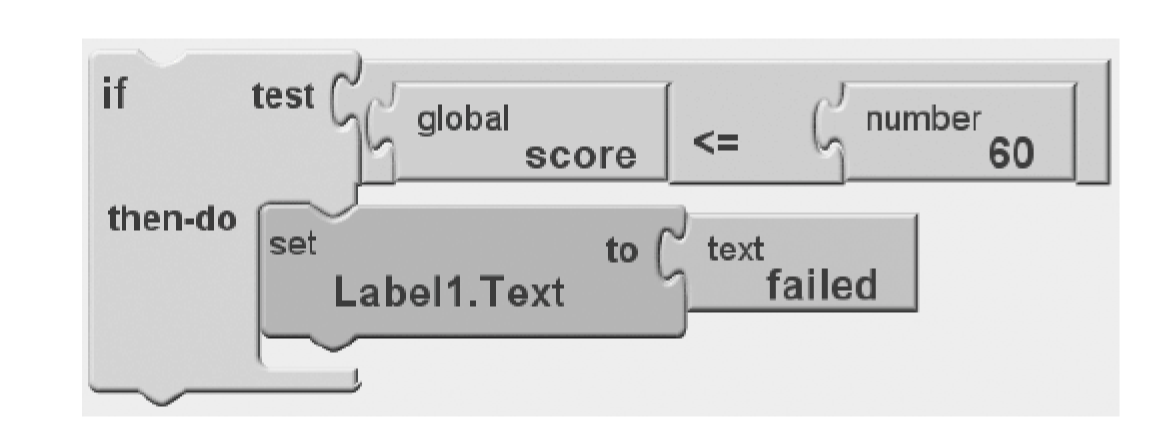


图 4.4 if 模块示例

4.2.2 布尔表达式

4.2.1 节中经常提到拼接在槽 test 上的“条件”,实际上是一种布尔表达式,最终只有 true(真)和 false(假)两个取值。

最简单的布尔表达式是等式(equality),这种布尔表达式用来测试一个值是否与另一个值相同。它可以是一个简单的等式,例如 2=4,也可以是一个复杂一些的等式,例如 score=60,如图 4.5 所示。

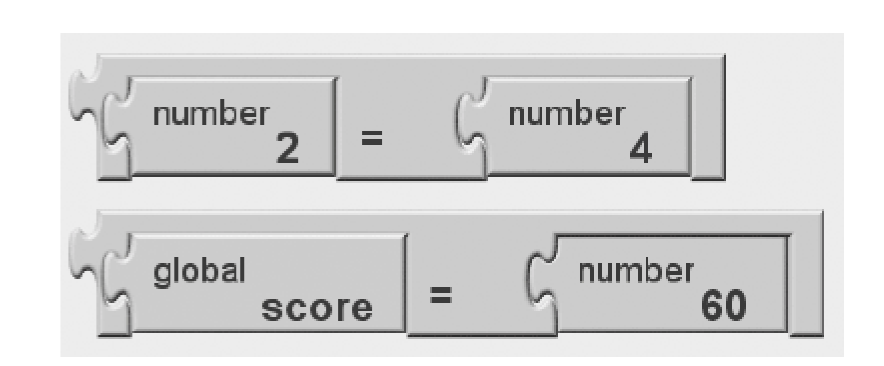


图 4.5　等式

App Inventor 支持的关系运算符有小于(<)、小于等于(<=)、等于(=)、大于(>)、大于等于(>=)、不等于(not=),如图 4.6 所示。使用这些关系运算符就可以做出一些较为复杂的条件判断。

App Inventor 支持的逻辑运算符有真(true)、假(false)、与(and)、或(or)和非(not),如图 4.7 所示。

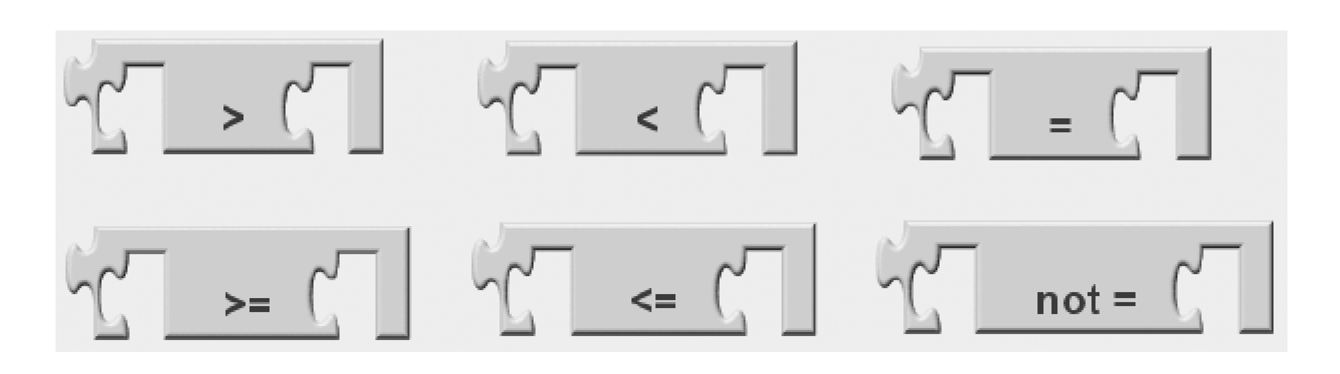



图 4.6　App Inventor 支持的关系运算符

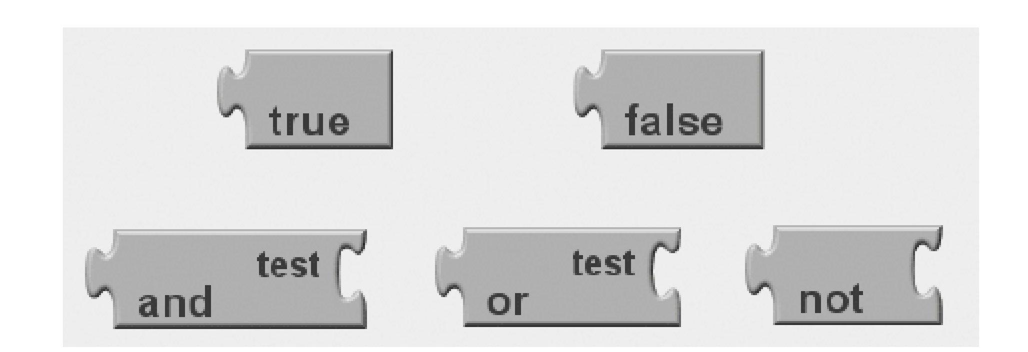


图 4.7　App Inventor 支持的逻辑运算符

通过使用关系运算符和逻辑运算符,可以表达稍微复杂一些的判断条件,例如图 4.8 的"高危工种的招聘条件是:考试分数大于等于 60 分,年龄小于等于 28 岁,初级工作在 5 年以上或高级工作在 2 年以上,而且必须是未婚"。使用到了逻辑运算符 and、or 和 not,以及像大于、小于等的一些关系运算符。

除此之外,还有一些判断模块返回的值同样是布尔类型的,这些模块也可以作为布尔表达式拼接在 if 模块的槽 test 上,如图 4.9 所示。通过观察,这些模块有一个共同的特点,都是"is…?"的形式,所以读者也可以较为容易地在众多的模块中将其辨认出来。

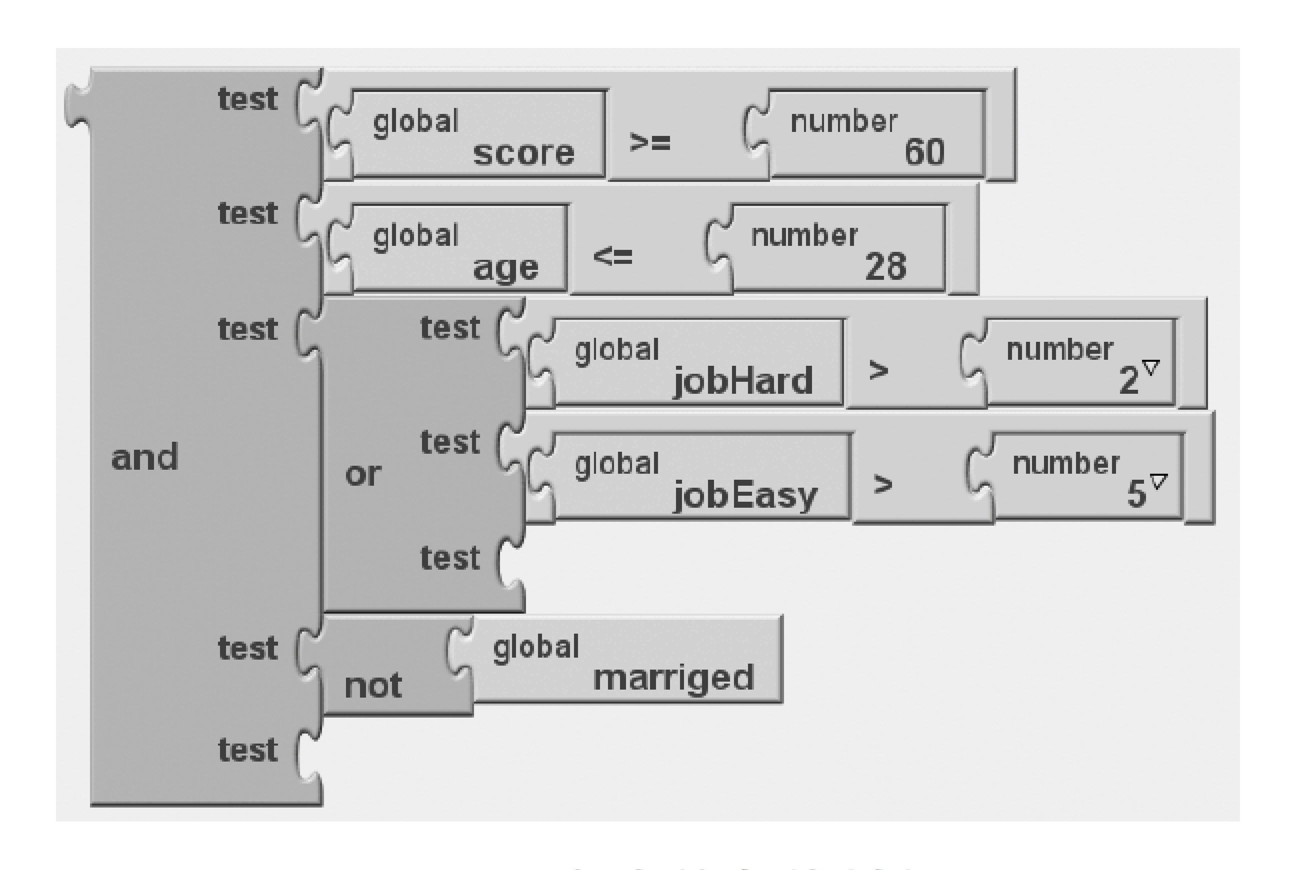


图 4.8　复杂的条件判断

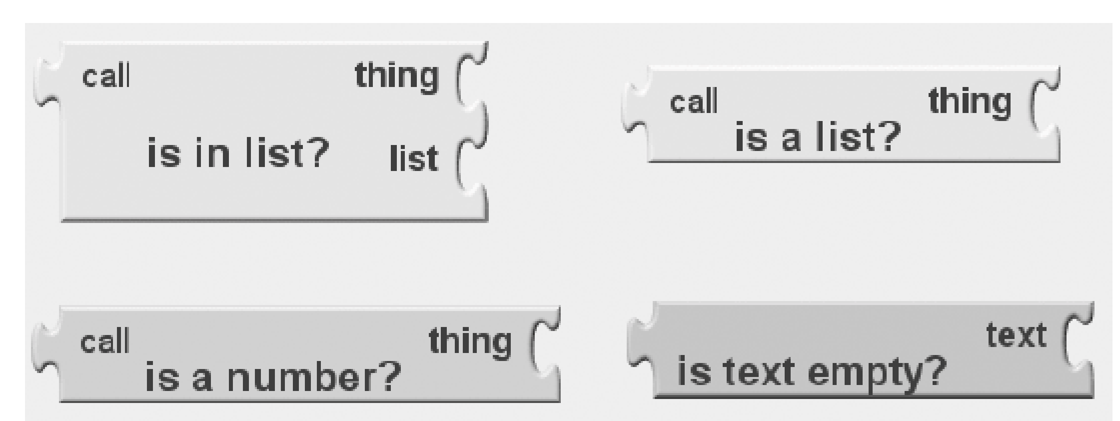


图 4.9　可用于条件判断的模块

4.2.3　ifelse 模块

ifelse 模块的结构与 if 模块相似,槽 test 用来拼接条件,槽 then-do 用来拼接需要执行的动作。不同之处在于多了一个槽(else-do),可以执行一些条件不成立时的操作。

ifelse 模块的执行流程是先判断槽 test 的条件是否为 true(条件成立),如果条件成立,则去执行槽 then-do 中的模块;如果条件为 false(条件不成立),则执行 else-do 中的模

块。ifelse 模块与执行流程如图 4.10 所示。

ifelse 模块给出一个与 if 模块相似的示例，不同之处是“当成绩大于 60 分时，显示考试成功的提示信息”，如图 4.11 所示。

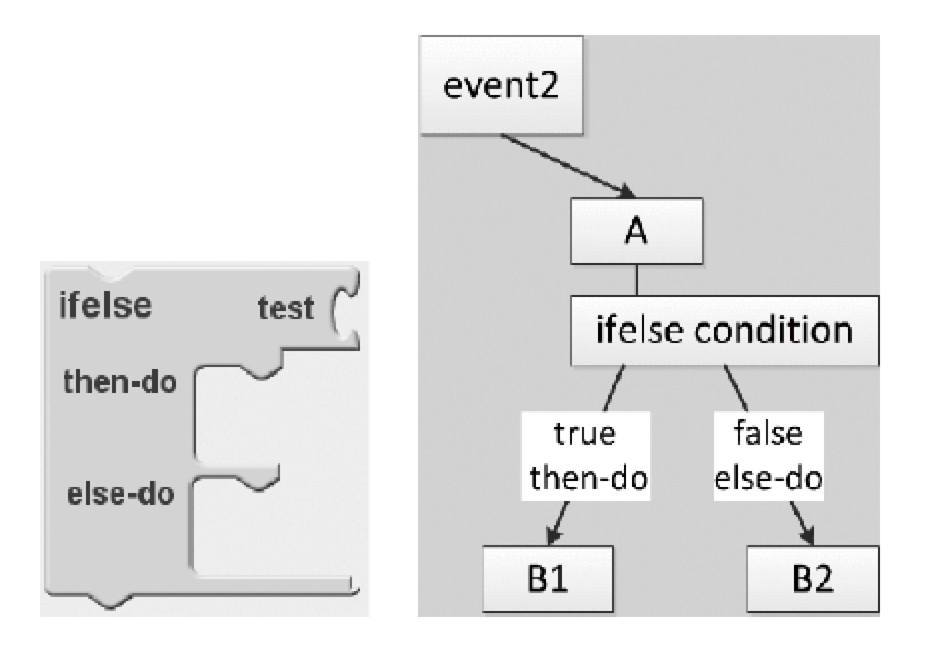


图 4.10 ifelse 模块与执行流程

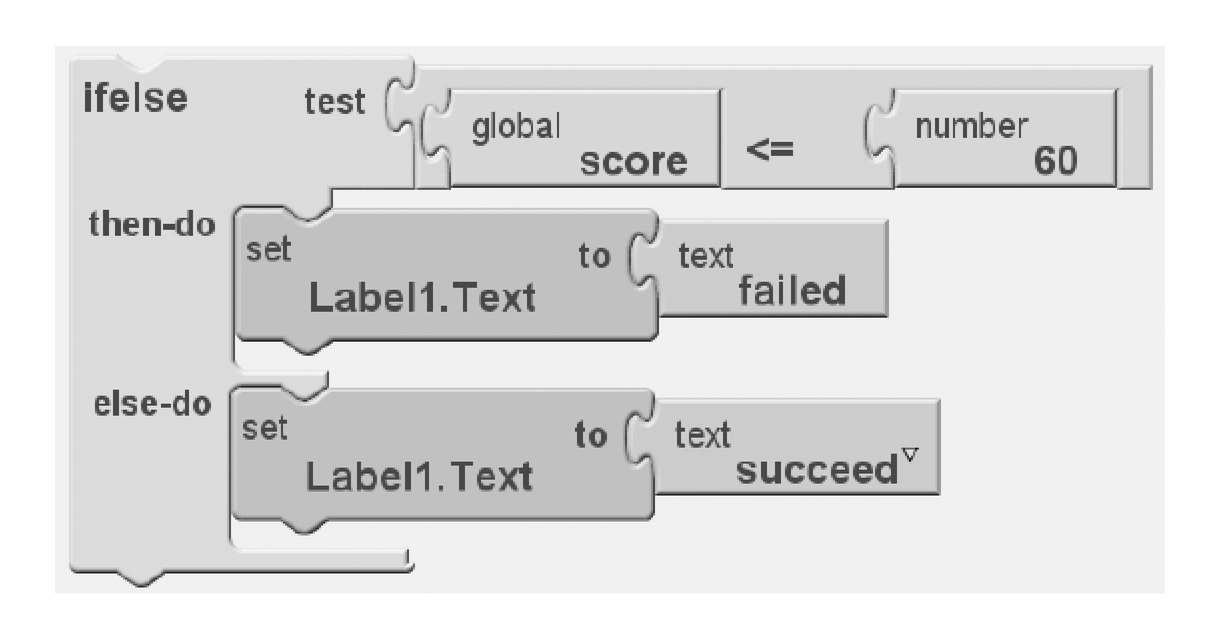


图 4.11 ifelse 模块示例

ifelse 模块示例首先判断槽 test 的条件“score<=60”是否成立，如果条件成立，则执行 then-do 中的内容，将 Label1 标签的显示内容修改为 failed。如果条件不成立，则执行 else-do 中的内容，将 Label1 标签的显示内容修改为 succeed。

4.2.4 条件嵌套

if 模块和 ifelse 模块可以单独使用，也可以嵌套使用。当程序中的条件分支多于两个时，则可在条件模块的 then-do 分支或 else-do 分支中嵌套 if 模块或 ifelse 模块，以实现多分支的情况。

下面用一个小例子说明如何使用条件嵌套。给出一个变量 x，x 可能的值只有 1、2 和 3，如果 x 为 1，显示“x=1”；如果 x 为 2，显示“x=2”；如果 x 为 3，显示“x=3”；如果 x 为其他值，则显示 error。在图 4.12 中，实现了这个例子的 4 条条件分支，使用了 ifelse 模块的嵌套结构。

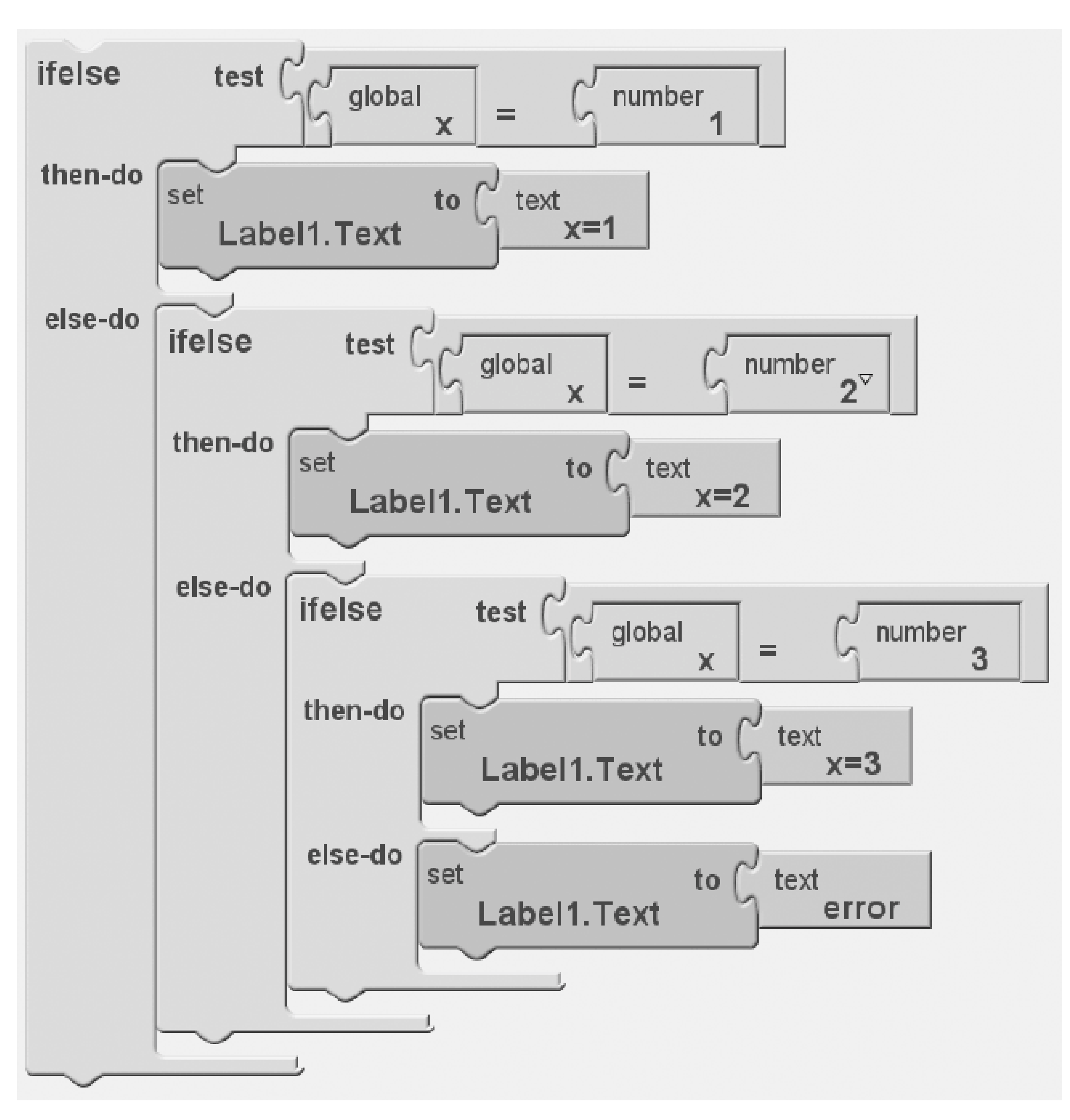


图 4.12 ifelse 模块的嵌套结构

上面的示例充分展现了条件嵌套的强大功能，能根据需要进行多层嵌套，以实现高复杂度的条件判断。但在使用条件嵌套结构时，要仔细分析程序的各分支情况，以免造成结构的混乱。

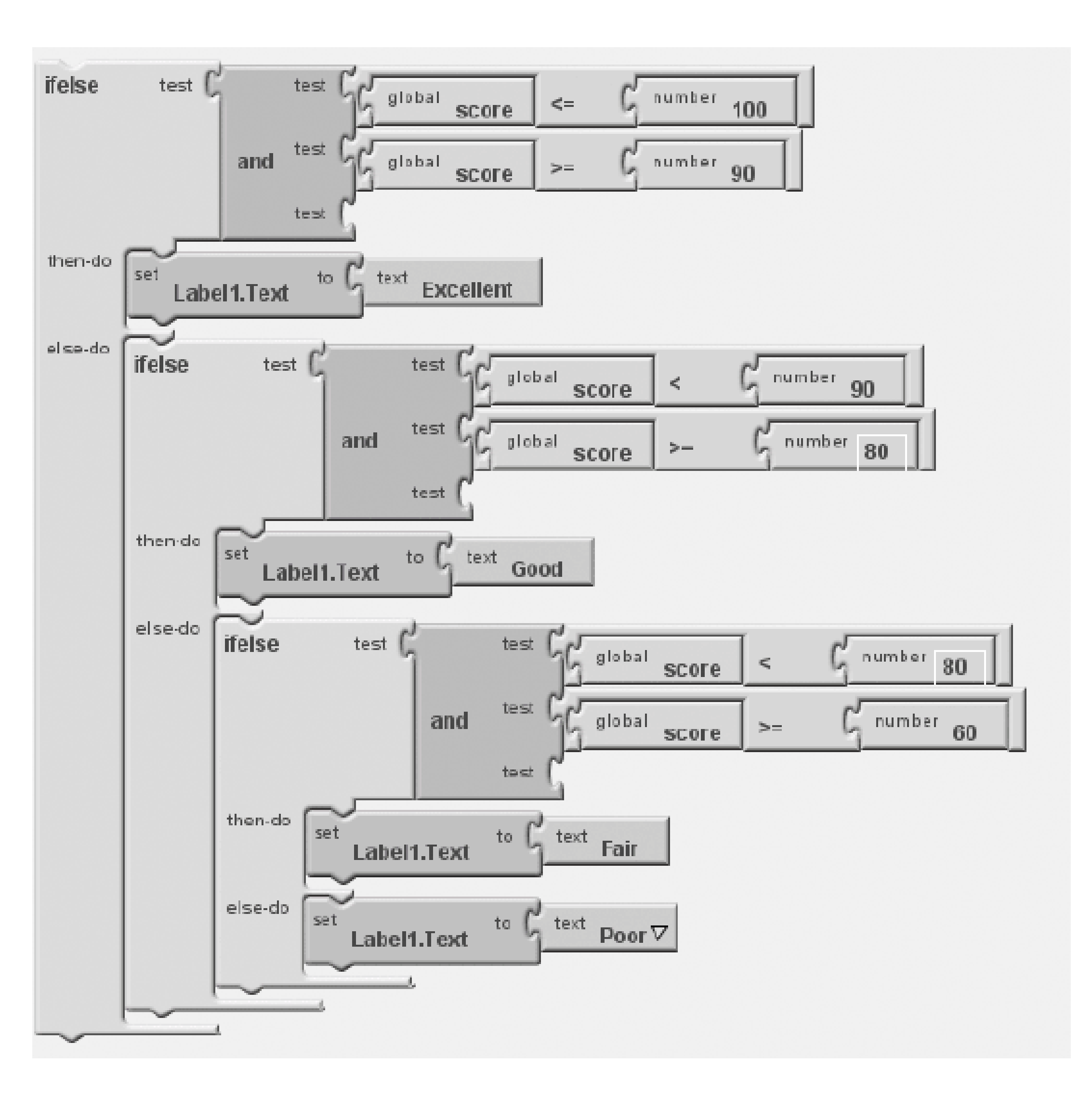


图 4.13　多条件判断和条件嵌套示例

在程序设计中，经常会遇到同时使用条件嵌套和多条件判断的情况。多条件判断是在使用 if 模块或 ifelse 模块时，在槽 test 中拼接多个条件，对多个条件进行综合判断，才能够返回一个最终的结果。

多条件判断一般要使用逻辑运算符 and 或 or。在需要多个条件同时成立时，使用逻辑运算符 and。如果仅需要多个条件中，至少其中一个条件成立时，使用逻辑运算符 or。

下面介绍一个小示例，用以说明如何使用多条件判断和条件嵌套，如图 4.13 所示。在这个示例中，通过判断 score 的值，确定成绩的类型：score 的值在 90～100 之间，成绩类型为 Excellent；在 80～90 之间，成绩类型为 Good；在 60～80 之间，成绩类型为 Fair；在 0～60 之间，成绩类型为 Poor。

该示例很好地呈现了多条件判断与条件嵌套相结合的程序结构，表面上程序中的条件嵌套重重，但仔细分析其结构后，会发现该程序脉络清晰，易于理解。

4.3　列　表

在程序设计过程中，数据是要处理的对象，也是需要展示的部分。程序中的数据是多样化的，数据的结构也比较复杂，为了处理这些复杂的数据结构，App Inventor 提供了“列表”来处理像电话册、购物清单这样的数据，并提供了关于列表的操作模块，如列表创建模块、列表项添加模块和列表项选择模块。

列表是将数据按照特定顺序进行排列的一种数据结构。列表中的每一个数据都有位置信息，称为索引，用户可以根据索引找到列表中与之对应的数据。

严格地讲，App Inventor 所有建立的列表都是动态的，也就是说，列表在建立后随时可以向列表中添加或删除数据。为了表述清晰，本书将 App Inventor 的列表分为静态列表和动态列表。静态列表在初始化时添加数据，在程序运行期间不做任何修改。动态列表在初始化时不添加数据，而是在程序运行期间动态地添加或删除数据。

4.3.1　静态列表

列表一般都保存在一个变量中，所以在介绍列表的建立方法前，先来说明一下如何创建变量。

创建变量需要使用 variable 模块，建立用来保存列表的全局变量，可以通过选择 Built-In→Definition→variable 找到 variable 模块，如图 4.14 所示。

然后，将 variable 模块拖曳到模块编辑器中。因为 variable 模块的名称不能够重复，且应具有明确的含义，因此下一步为列表变量修改名称。将 variable 模块的名称由 variable 改为 phonenumbers，如图 4.15 所示。

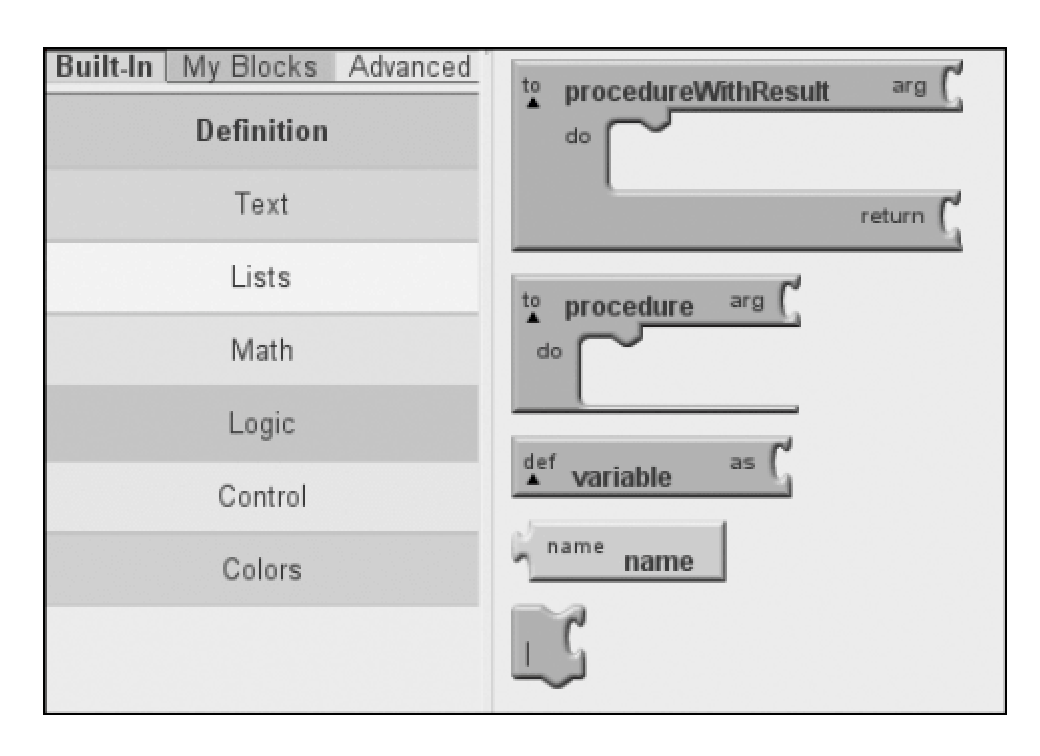


图 4.14 模块编辑器中的 variable 模块

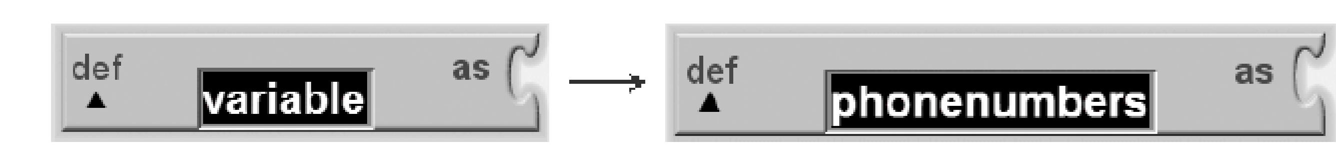


图 4.15 修改 variable 模块名称

接下来，选择类表创建模块 make a list，方法是执行 Built-In→Lists→make a list 菜单命令，如图 4.16 所示。

然后将 make a list 模块与 variable 模块(phonenumbers 变量)拼接在一起，表示 phonenumbers 变量是一个列表变量，如图 4.17 所示。

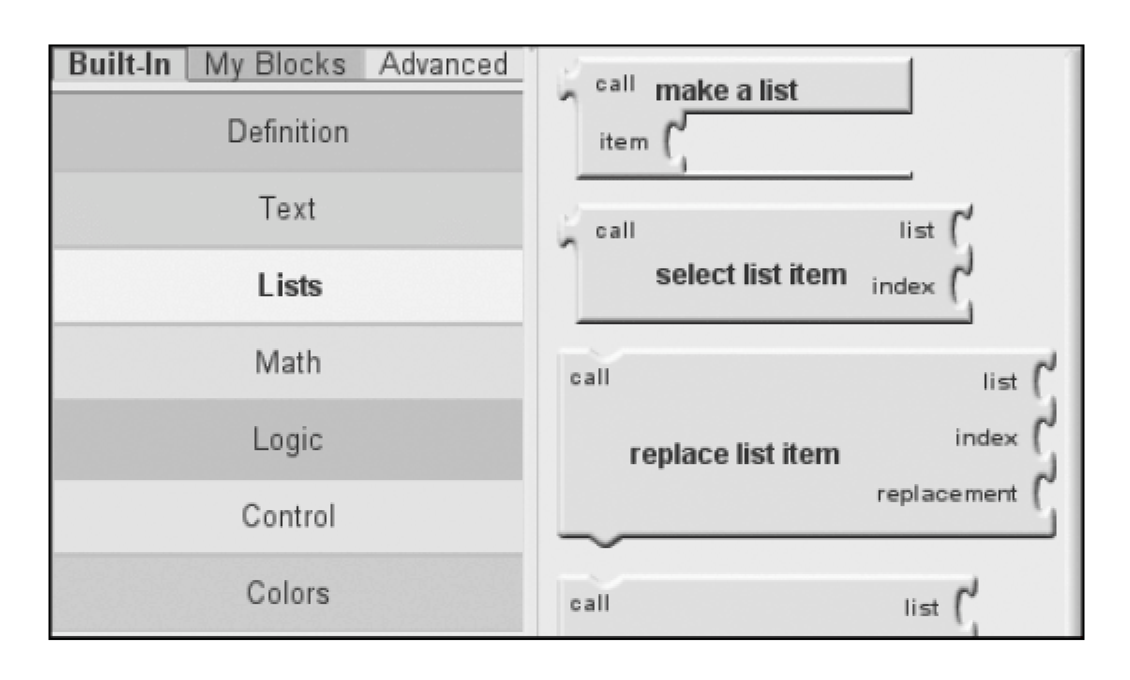


图 4.16 模块编辑器中 make a list 模块

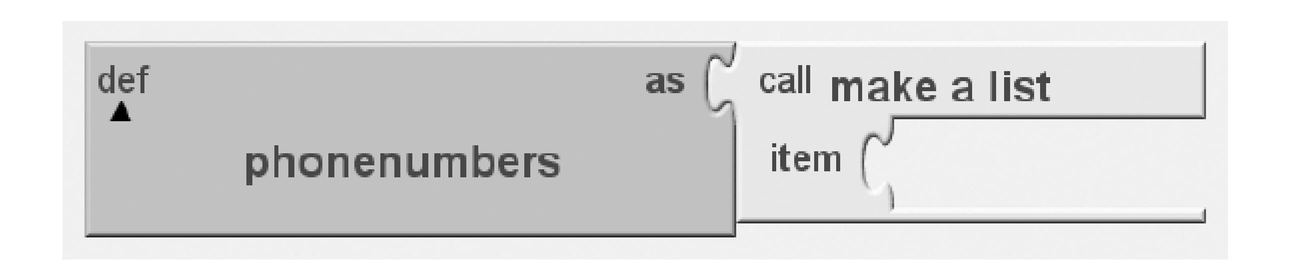


图 4.17 列表变量创建完成

只调用 make a list 模块，而没有在槽 item 中添加任何元素，则只是创建了一个空列表，如图 4.18 所示。

静态列表一般在调用 make a list 模块进行初始化的时候添加元素，这样只要在槽 item 中添加需要的信息即可。这里将文本"1010101"和"2020202"拼接到槽 item 上，如图 4.19 所示。随着向槽 item 拼接的元素数量的增加，槽 item 的数量总会比元素的数量多 1 个，这样总是有一个空着的槽 item 可以拼接下一个列表元素。



图 4.18 make a list 模块

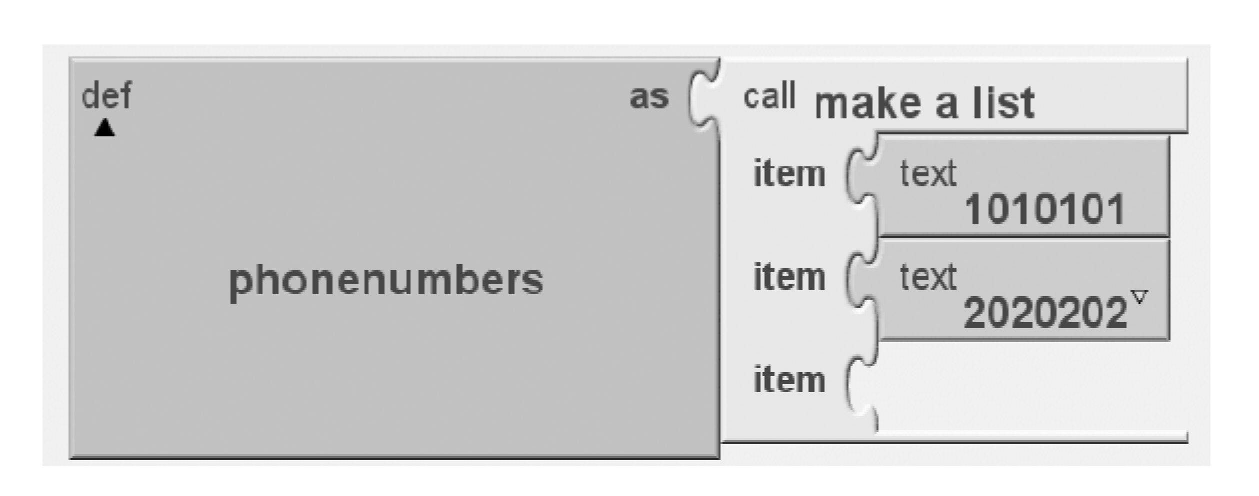


图 4.19 列表数据填充

此时的 phonenumbers 列表则包含两个元素，索引为 1 的元素是文本“1010101”，索引为 2 的元素是文本“2020202”。需要注意的是，App Inventor 的列表索引是从 1 开始的。

4.3.2 获取列表项

4.3.1 节定义了列表变量 phonenumbers，这样就可以在需要使用时随时找到这个列表。在需要对列表进行操作时，只要找到调用包含该列表的全局变量即可。

对列表最常用的操作就是获取列表项，一般获取列表项是通过索引实现的。在 4.3.1 节的列表 phonenumber 中，包含两个文本元素“1010101”和“2020202”，下面将说明如何通过使用 select list item 模块，获取列表 phonenumber 中索引为 2 的元素“2020202”。

首先从 Built-In→Lists→select list item 路径中找到 select list item 模块，如图 4.20 所示。

select list item 模块中的参数槽有两个，分别是 list 和 index，如图 4.21 所示。槽 list 用来定义被操作的目标列表，槽 index 用来定义索引编号。

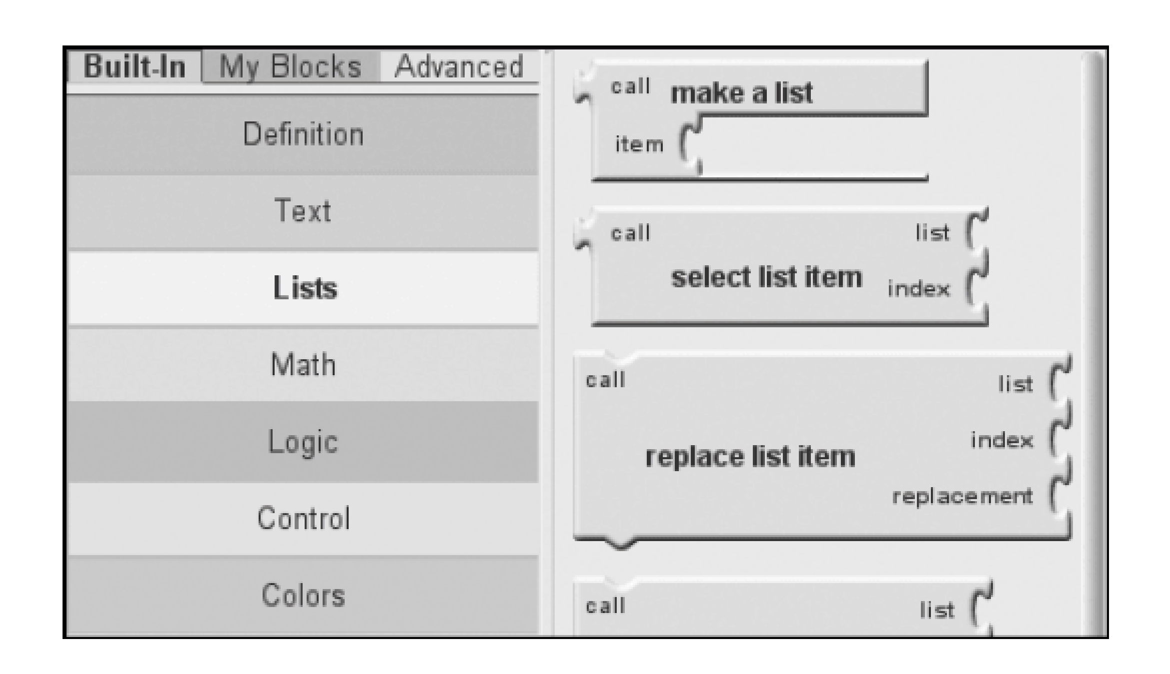


图 4.20 模块编辑器中 select list item 模块

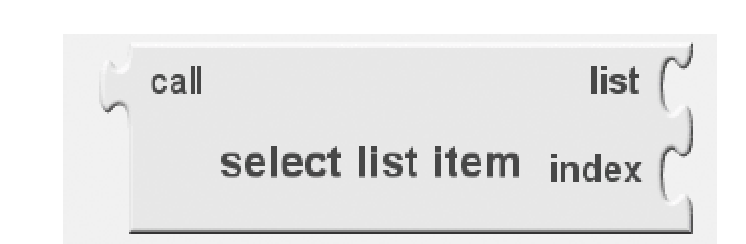


图 4.21 select list item 模块

将列表变量 phonenumbers 拼接在槽 list 上，将数字 2 拼接在槽 index 上，如图 4.22 所示。在 phonenumbers 列表中，获取索引号为 2 的元素。select list item 模块运行后的返回值是文本“2020202”。

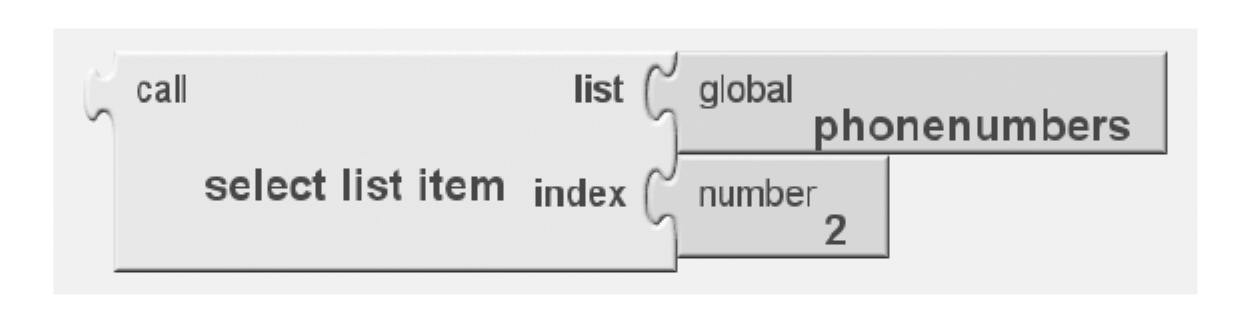


图 4.22 获取 phonenumbers 列表索引号为 2 的元素

4.3.3 遍历列表

前面介绍了获得列表中某一元素的方法，但在实际应用中，对列表所有元素的操作却经常被使用到，例如，显示列表中所有的元素、在列表中找到最小的元素、累加列表中所有的元素等。

App Inventor 一般利用列表的索引对列表中所有元素进行遍历。具体的实现方法是先自定义变量 index 表示列表索引，并将变量 index 的初始值设为 1，如图 4.23 所示。

创建索引变量后，便可根据索引变量 index 找到列表的数据项，若将索引变量初始值设为 1，并递增索引变量，就可以依次找到列表的第一项、第二项、第三项……，从而实现了对列表的遍历。

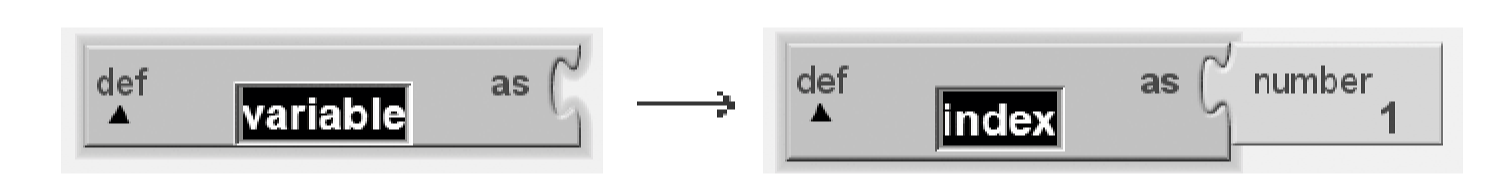


图 4.23　初始化列表索引变量 index

选择 phonenumbers 列表下一项的模块结构如图 4.24 所示。

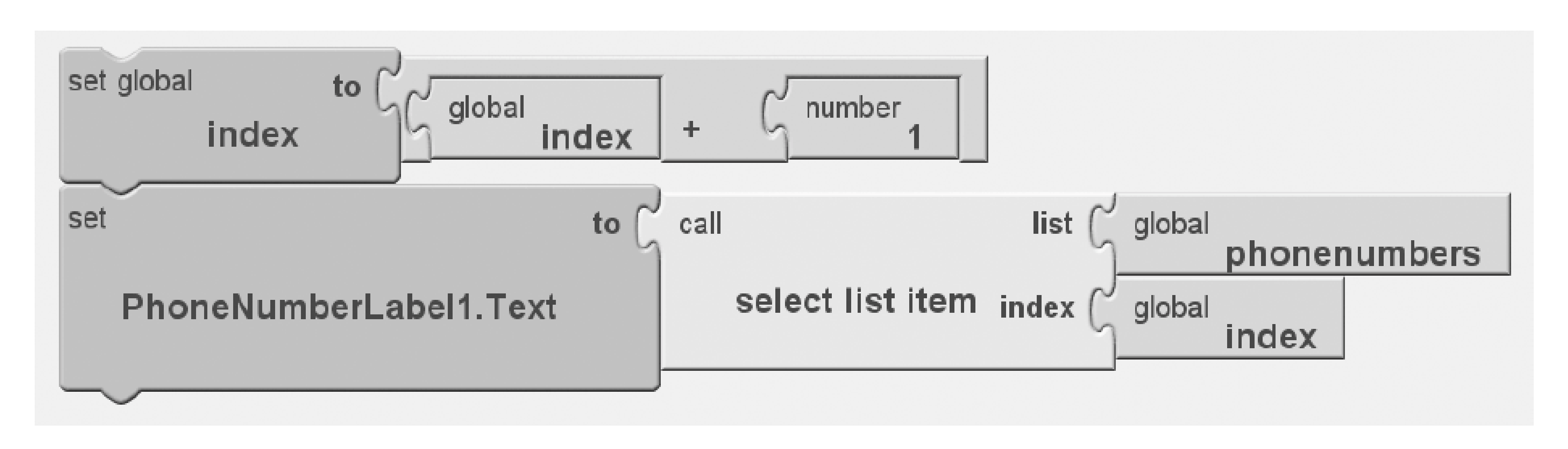


图 4.24　选择列表的下一项

在遍历整个列表的过程中，首先实现索引变量 index 加 1 递增，然后使用这个索引变量 index 获取列表中的元素。

下面介绍一个小示例 Number，用来说明如何自定义索引，通过索引遍历列表，以及避免索引超出列表范围的方法。Number 示例的运行界面如图 4.25 所示。

第一步先建立列表变量 phoneNum，列表中包含两个元素"1010101"和"2020202"，如图 4.26 所示。



图 4.25　Number 示例的运行界面

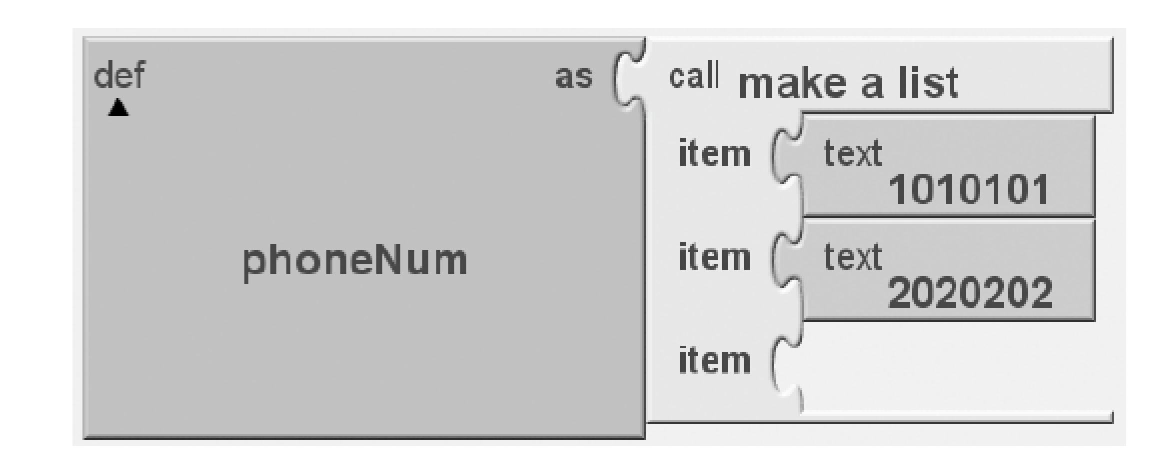


图 4.26　初始化列表变量

在图 4.27 中，全局变量 index 的初始化值为 1，当第一次触发 Button1.Click 事件时，索引变量 index 指向列表的第一项位置，即"1010101"，并将变量 index 累加 1，index 的值变为 2。在第二次触发 Button1.Click 事件时，此时的 index 值为 2，索引变量 index 指向

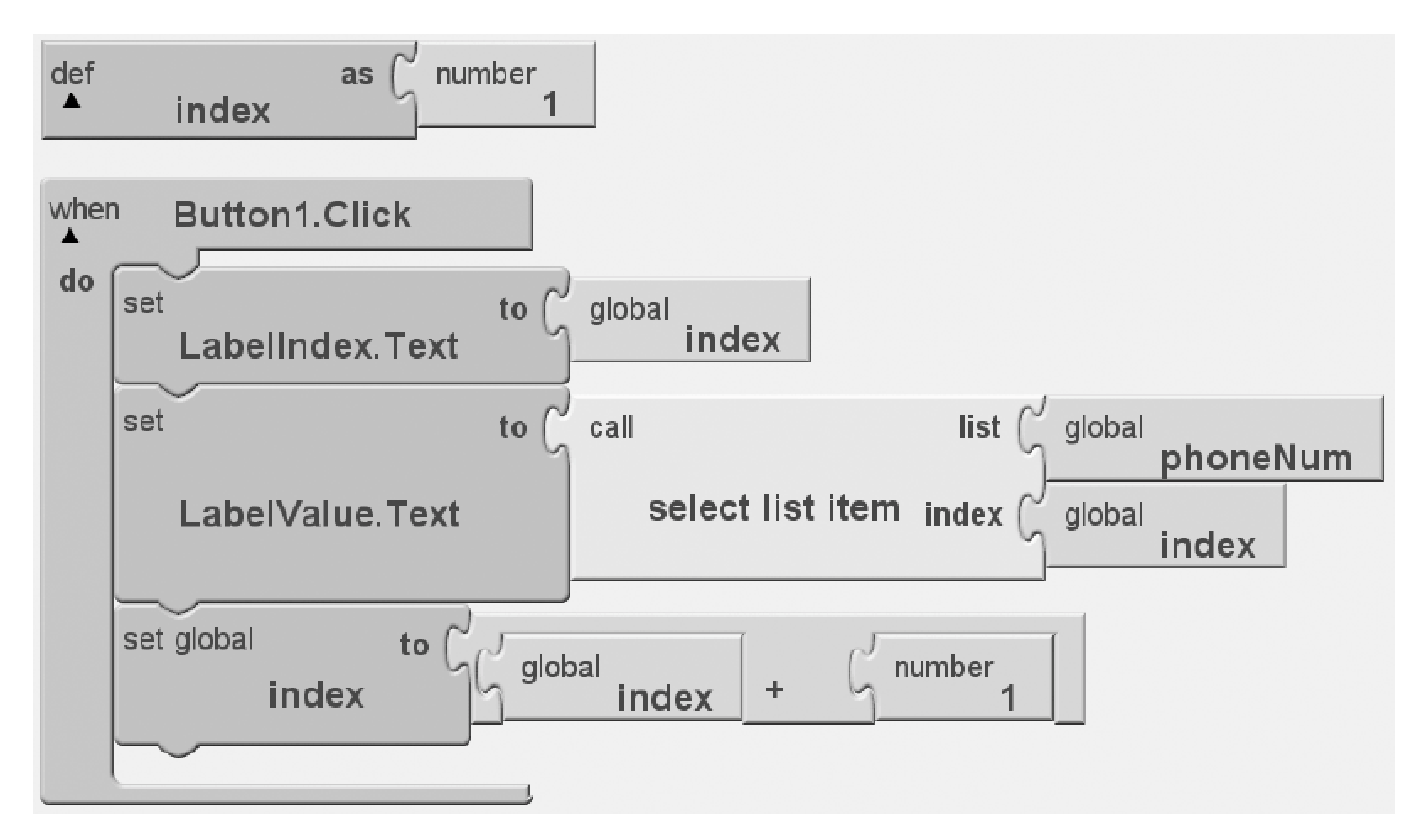


图 4.27　列表遍历模块

列表的第二项位置，即“2020202”，并将变量 index 再次累加 1，index 的值变为 3。

当第三次单击按钮触发 Button1. Click 事件时，异常出现了，如图 4.28 所示。出现异常的原因是索引变量 index 的值超出了列表项的数目，index 的当前值为 3，而索引只有两个元素。

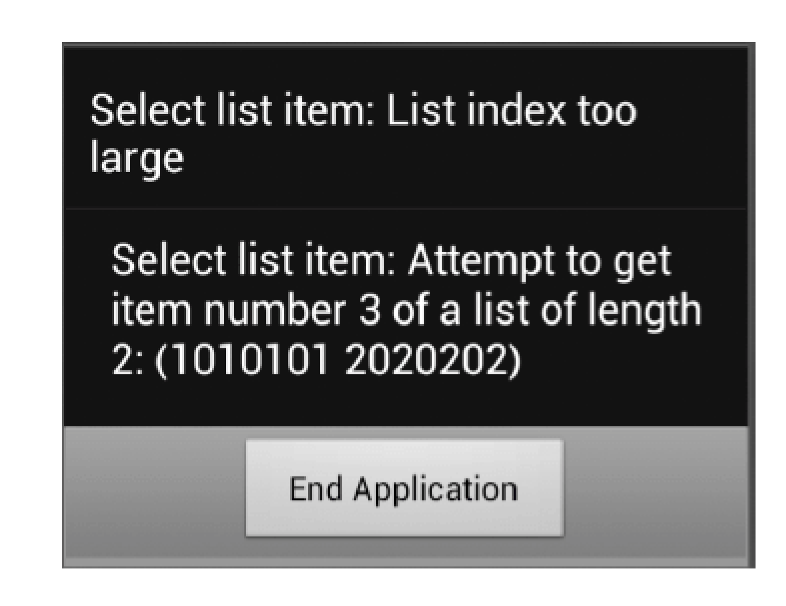


图 4.28　索引超范围所引发的异常

为了解决这个问题，需要在使用索引变量值前，检查索引变量是否超出列表长度，修改后的模块结构如图 4.29 所示。使用了条件判断模块 if，将变量 index 和列表变量 phoneNum 的长度(length of list)进行比较，如果变量 index 超出范围，就将变量 index 赋值为 1。

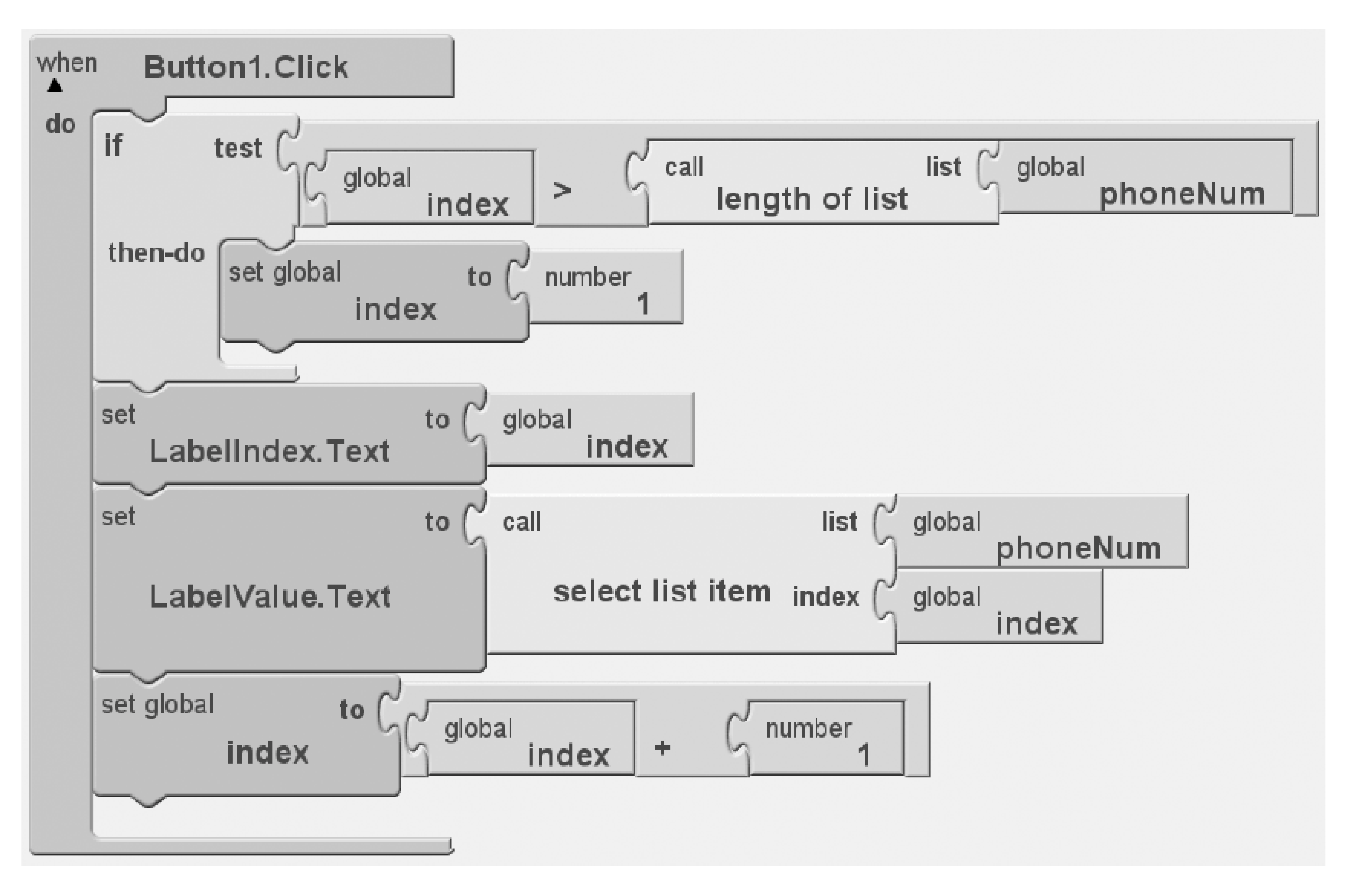


图 4.29　修改后列表遍历模块

下面给出 Number 示例的全部模块结构如图 4.30 所示。

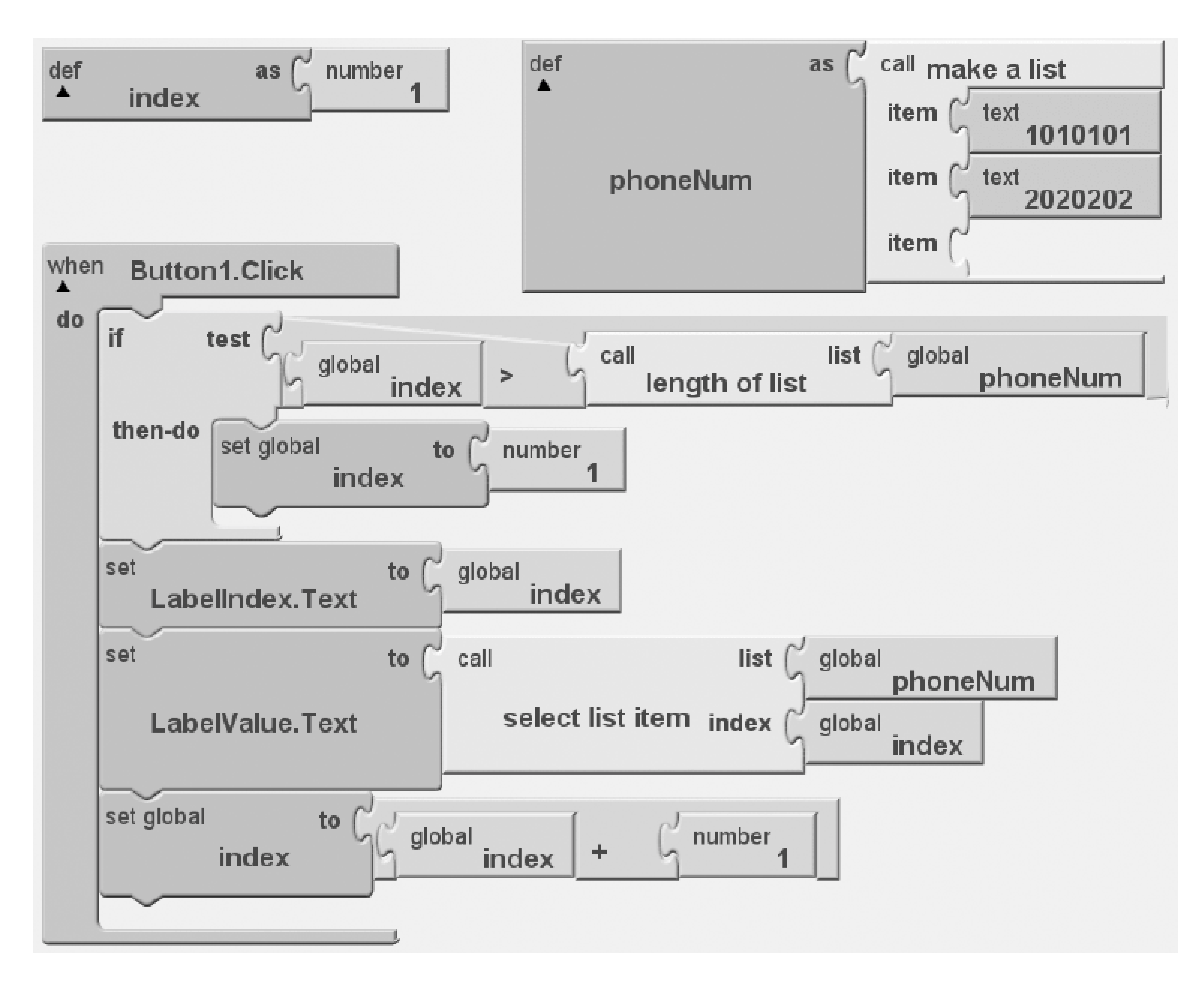


图 4.30　Number 示例的全部模块结构

4.3.4 动态列表

前面介绍了静态列表的创建方法,下面将介绍如何创建动态列表。

动态列表的创建方法和静态列表相似,不同之处是动态列表在创建时不需要初始化列表中的元素,也就是说在使用 make a list 模块的时候,不需要向槽 item 中添加元素。图 4.31 中创建了一个动态列表 phoneBook。

向动态列表中加入元素,需要调用 add items to list 模块,路径是 Built-In→Lists→add items to list,如图 4.32 所示。

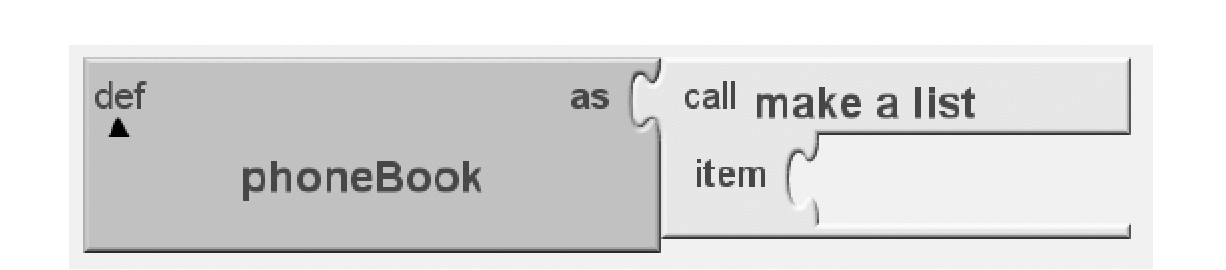


图 4.31 动态列表 phoneBook

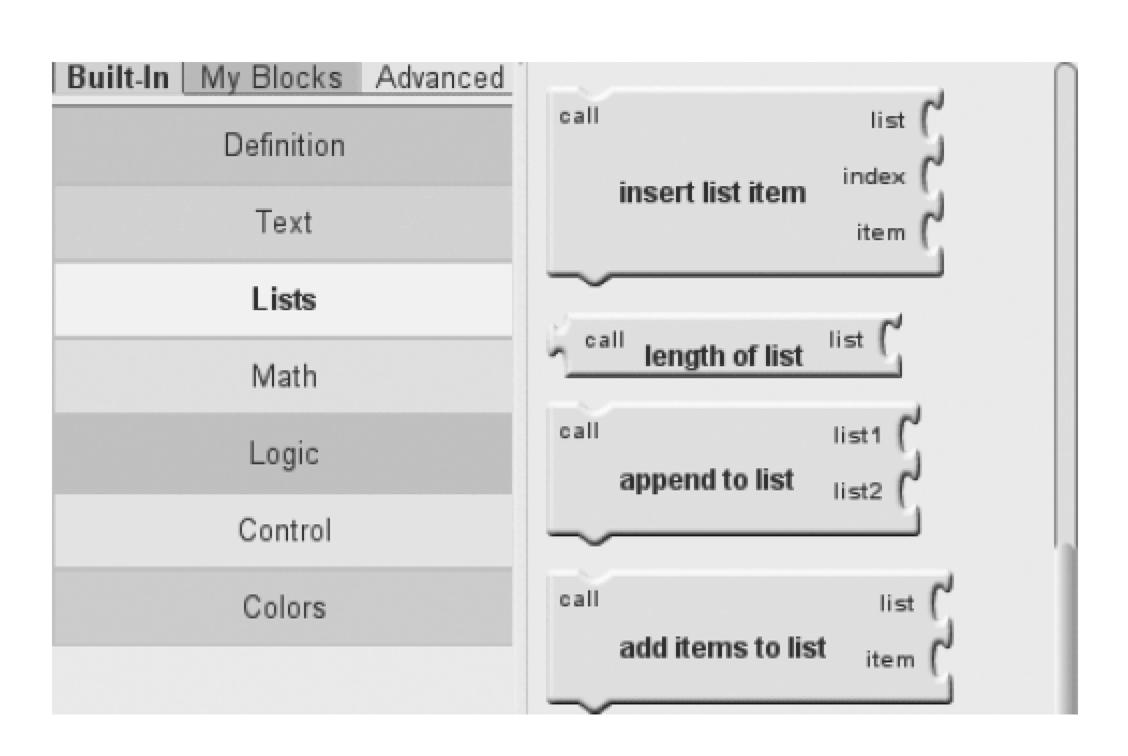


图 4.32 模块编辑器中 add items to list 模块

add items to list 模块中有两个参数槽,如图 4.33 所示。槽 list 用来选择目标列表对象,也就是要加入元素的列表变量,槽 item 是被添加到列表中的元素。

向动态列表中加入元素,只需要将列表变量拼接在 add items to list 模块的槽 list 上,将要加入的元素拼接在槽 item 上即可。图 4.34 是向列表 phoneBook 中添加了一个文本元素"1010101",从图中槽 item 的数量上可知,add items to list 模块支持向动态列表中一次加入多个元素。

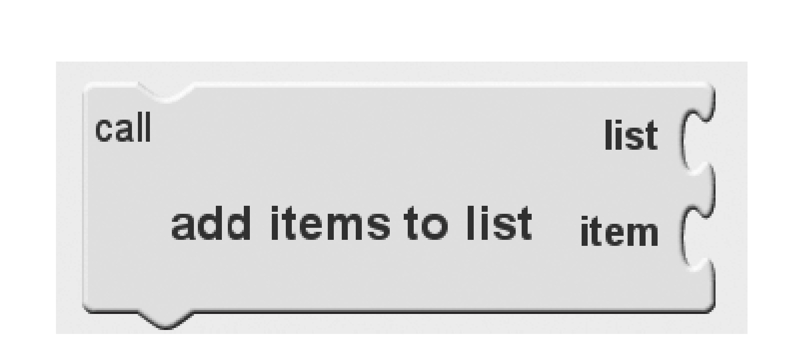


图 4.33 add items to list 模块

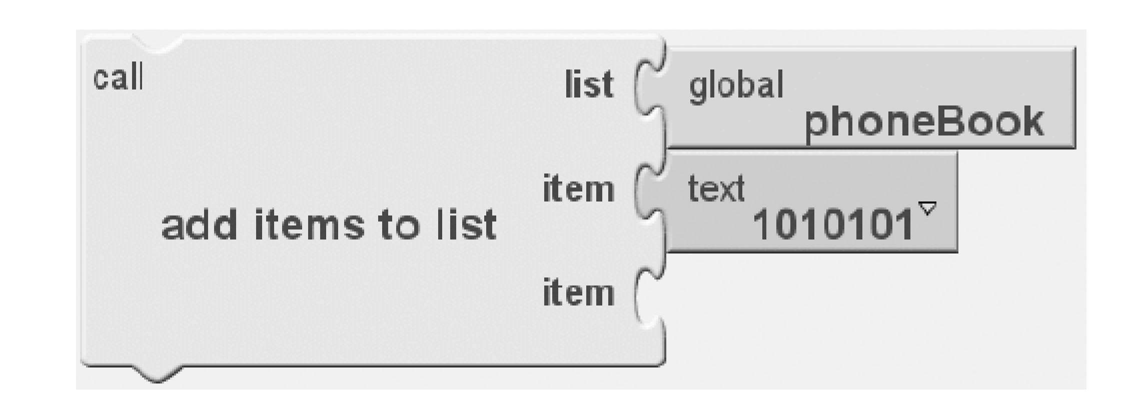


图 4.34 向动态列表中添加元素

从动态列表中删除元素,需要调用 remove list item 模块,路径是 Built-In→Lists→remove list item。remove list item 模块中也有两个参数槽,槽 list 表示目标列表对象,槽 index 表示要删除的元素在列表中的位置(索引),如图 4.35 所示。

如果要删除列表 phoneBook 中的第一项,则将数字 1 拼接在槽 index 上,将列表 phoneBook 拼接在槽 list 上,结构如图 4.36 所示。

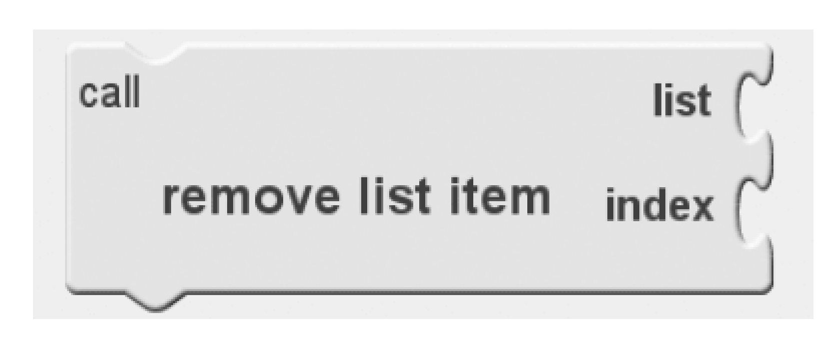


图 4.35 remove list item 模块

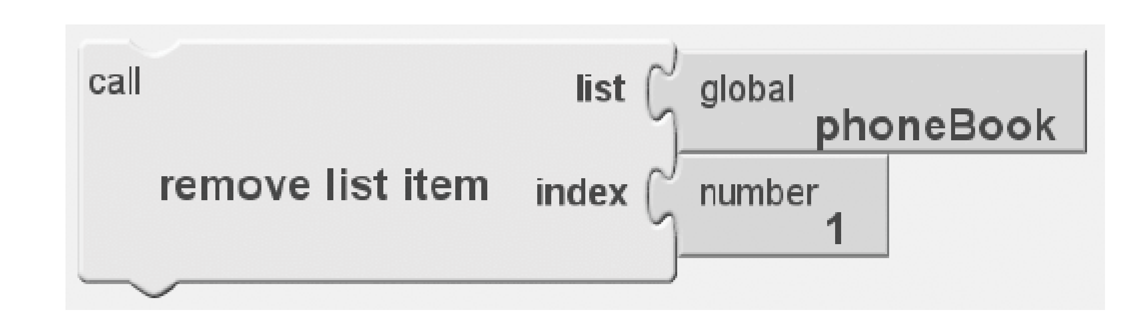


图 4.36 删除 phoneBook 列表中的第一项

但在多数情况下，用户还是希望通过元素删除列表中的项。还是以 phoneBook 为例，如果希望删除列表中的元素“1010101”，则可以调用 position in list 模块，路径是 Built-In→Lists→position in list。position in list 模块有两个参数槽，如图 4.37 所示，槽 thing 表示目标元素，槽 list 表示目标列表，该模块会返回目标元素在列表中的索引值。通过调用 remove list item 模块，根据索引值就可以删除用户想删除的列表元素了。

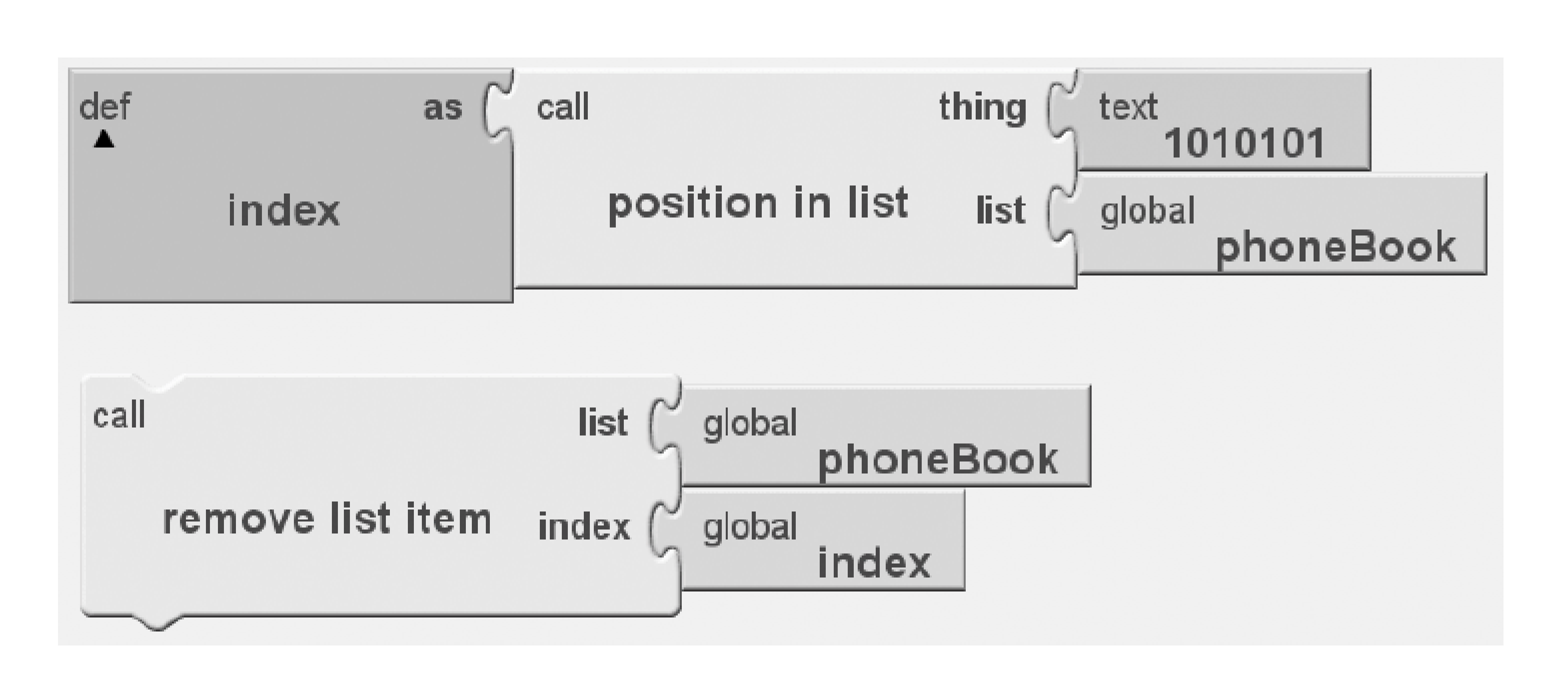


图 4.37 根据元素内容删除列表中的元素

4.3.5 列表嵌套

和条件判断的嵌套结构一样，列表同样支持嵌套结构。列表中的元素项可以是文本、数字、颜色，也可以是列表，这就是说，可以创建包含列表的列表。

下面来介绍如何实现列表嵌套，先创建两个静态列表 FruitList 和 AnimalList，每个列表中有两个列表元素，如图 4.38 所示。

然后建立列表变量 ListofList，调用 make a list 模块，将列表 FruitList 和 AnimalList 拼接到槽 item 上，完成列表嵌套结构，如图 4.39 所示。

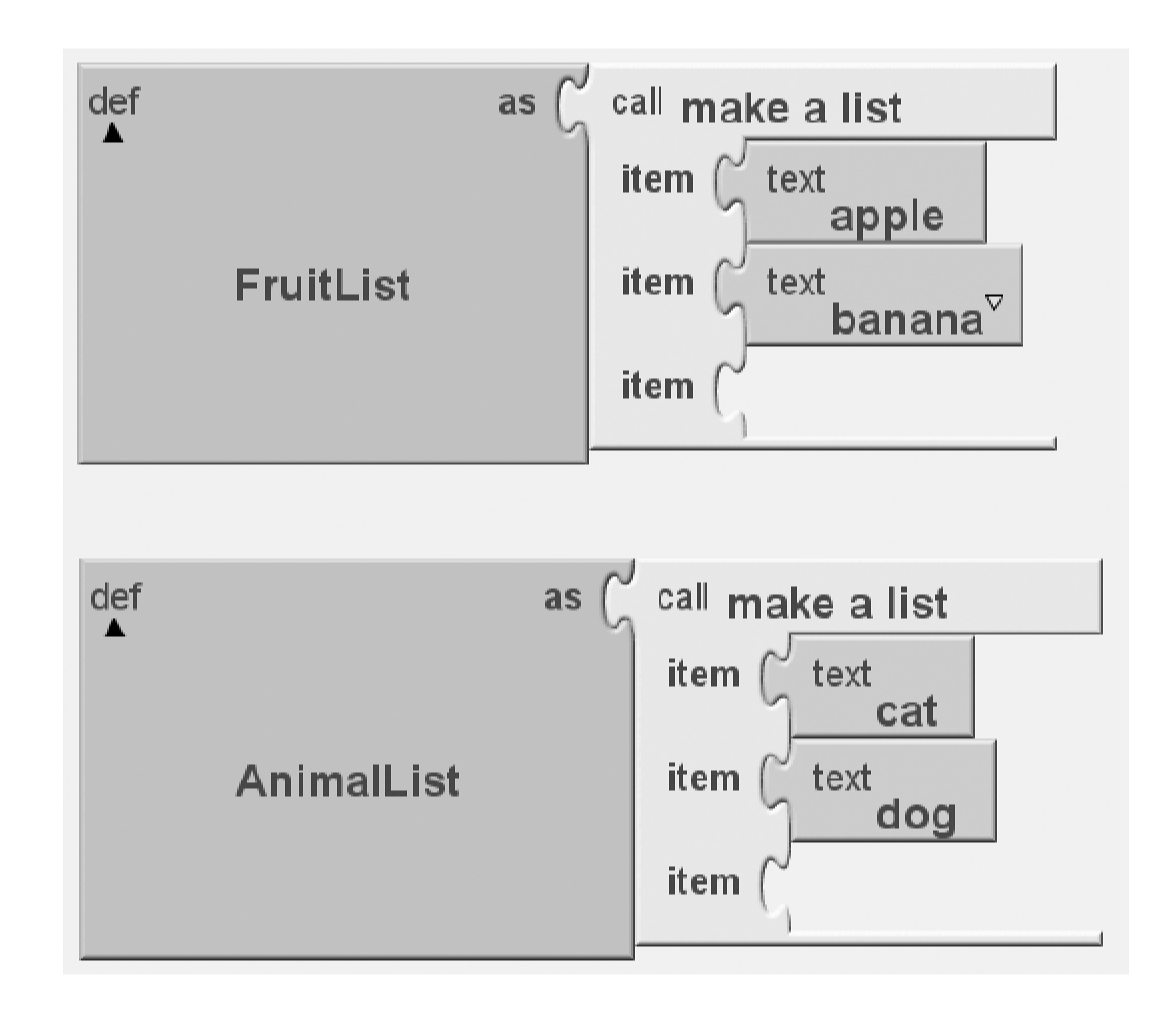


图 4.38 FruitList 和 AnimalList 列表

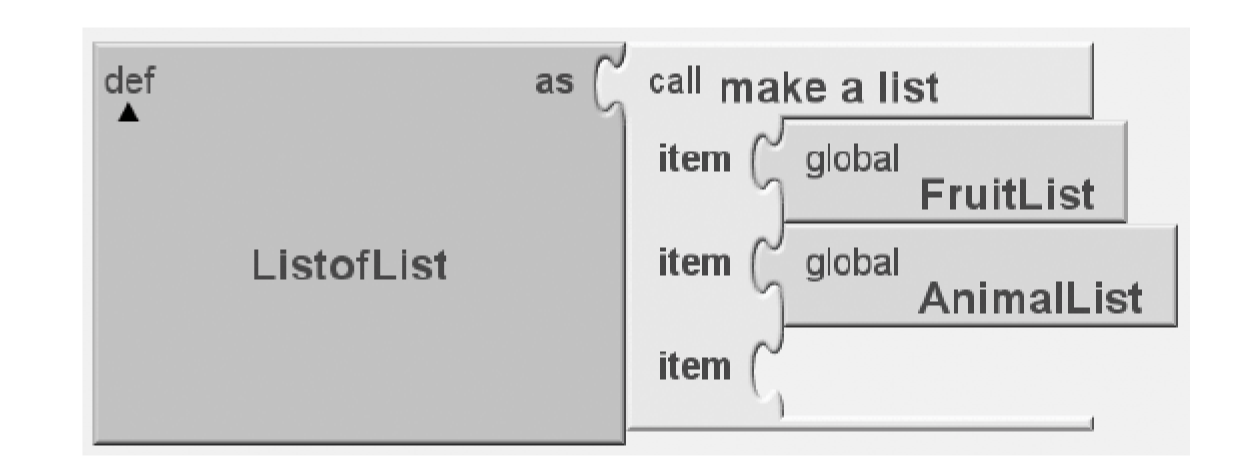


图 4.39 ListofList 列表

当然，也可以按照图 4.40 的方法实现列表嵌套结构，这种方法更加简洁。

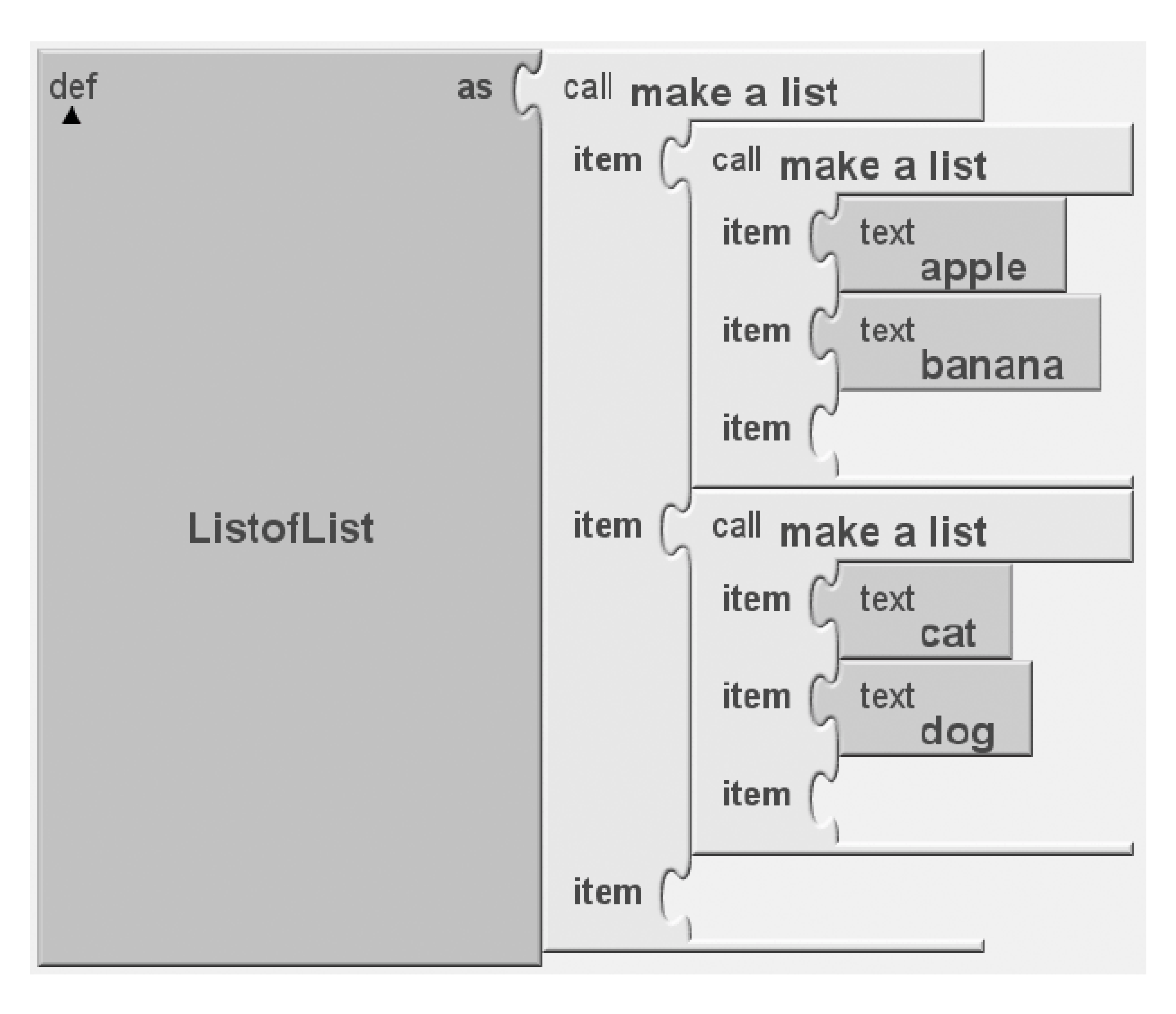


图 4.40　嵌套列表

4.4　循环结构

在程序设计过程中，循环也是一种经常使用到的结构。循环就是程序中重复执行同样的动作，这样的情况可以利用循环结构来实现。本节介绍两种循环结构 foreach 模块和 while 模块。

4.4.1　foreach

foreach 模块可以用来遍历列表，每次循环都获取列表中的一个元素。但需要注意的是，foreach 模块一次只能处理一个列表。foreach 模块的结构如图 4.41 所示，槽 in list 拼接需要遍历的列表，槽 variable 中的参数 var 是遍历列表过程中的元素。foreach 模块的路径为 Built-In→Control→foreach。

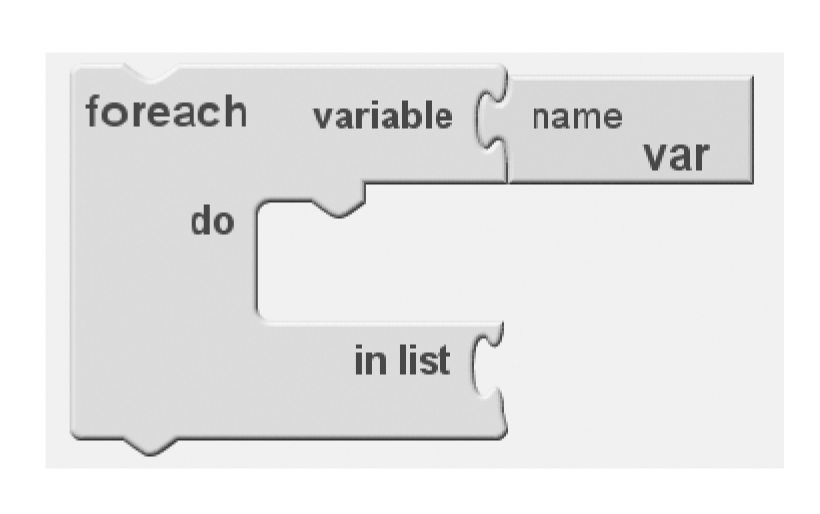


图 4.41　foreach 模块

在遍历过程中，foreach 模块首先从列表中获取第一个元素，赋值给变量 var，这样就可以在槽 do 中使用保存在 var 中的第一个列表中的元素。然后 foreach 模块再从列表中获取第二个元素，同样赋值给变量 var。如此反复，直到列表中最后一个元素被访问到，循环过程结束。

用一个累加数字的示例说明如何使用 foreach 模块。在这个示例中，列表 num 有三个数字元素：1、2 和 3，使用 foreach 模块遍历列表 num，将三个数字的累加值保存在全局变量 sum 中，如图 4.42 所示。

4.4.2　while

相对于 foreach 模块，while 模块则更加常用一些。while 模块是一种有条件循环结

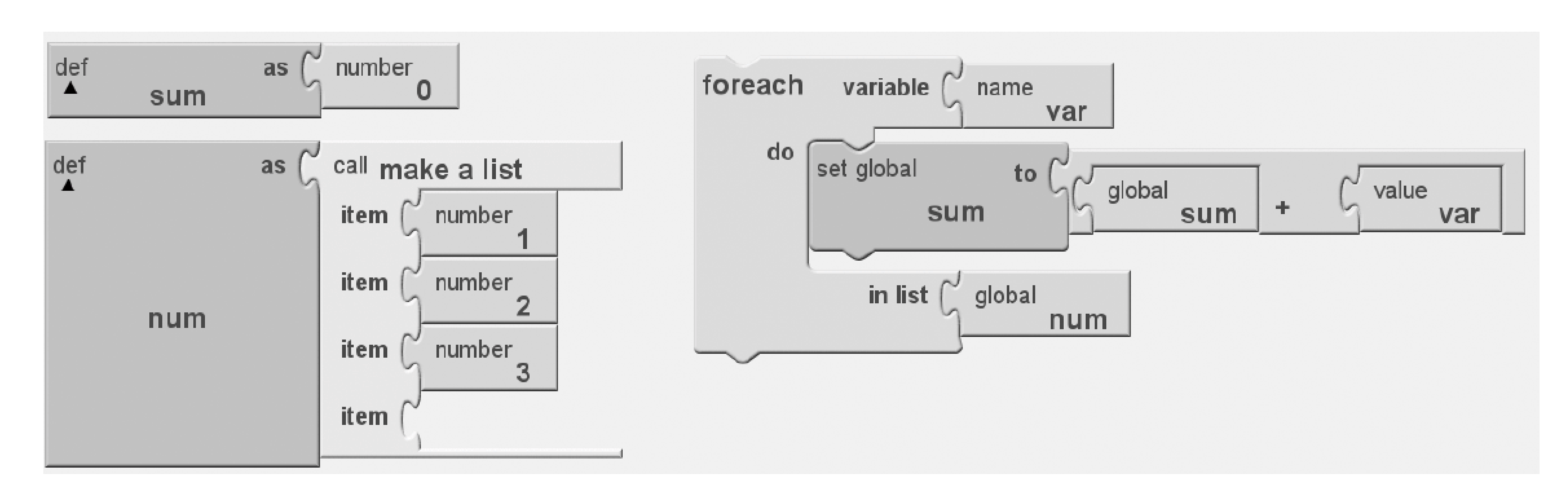


图 4.42　foreach 模块的累加数字示例

构，在条件满足时，循环过程将持续进行，当条件无法满足时，退出循环过程。

while 模块的结构如图 4.43 所示，槽 test 用来拼接判断 while 循环是否继续执行的条件语句，槽 do 是执行动作的循环部分。while 模块路径是 Built-In→Control→while。

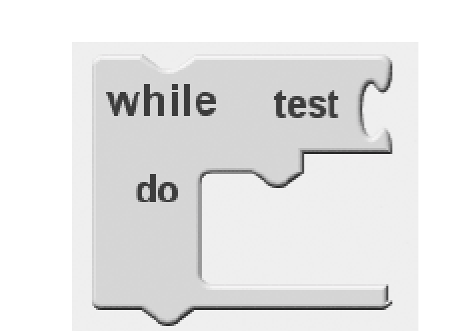


图 4.43　while 模块

while 模块里面的循环部分是否执行，要根据 test 中的条件进行判断，当 test 中的条件模块返回 true 时，程序执行槽 do 中的内容；若返回 false，则不执行槽 do 中的内容，而是继续执行 while 模块后面的程序模块。

while 模块的执行流程如图 4.44 所示。

还是以数字累加为例，说明如何使用 while 模块，如图 4.45 所示。在这个例子中，要获得数字 1～10 的累加结果。首先定义两个变量，变量 sum 表示累加值，变量 n 表示 1～10 的数字，初始值为 1。在 while 模块中，槽 test 的条件是“n≤10”，如果满足条件，槽 do 的动作是将变量 n 累加到 sum 变量中，并在每次循环过程中，将变量 n 增加 1。这样在 n 等于 1～10 期间，都满足条件“n≤10”，所以变量 sum 可以获取到 1～10 的累加值。当 n 等于 11 时，无法满足条件“n≤10”，while 模块结束循环。

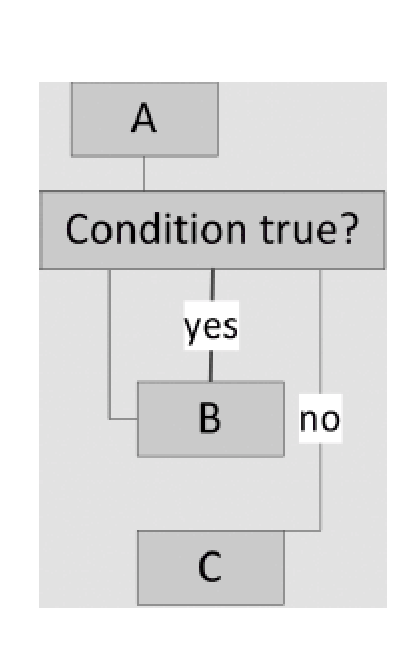


图 4.44　while 模块执行流程图

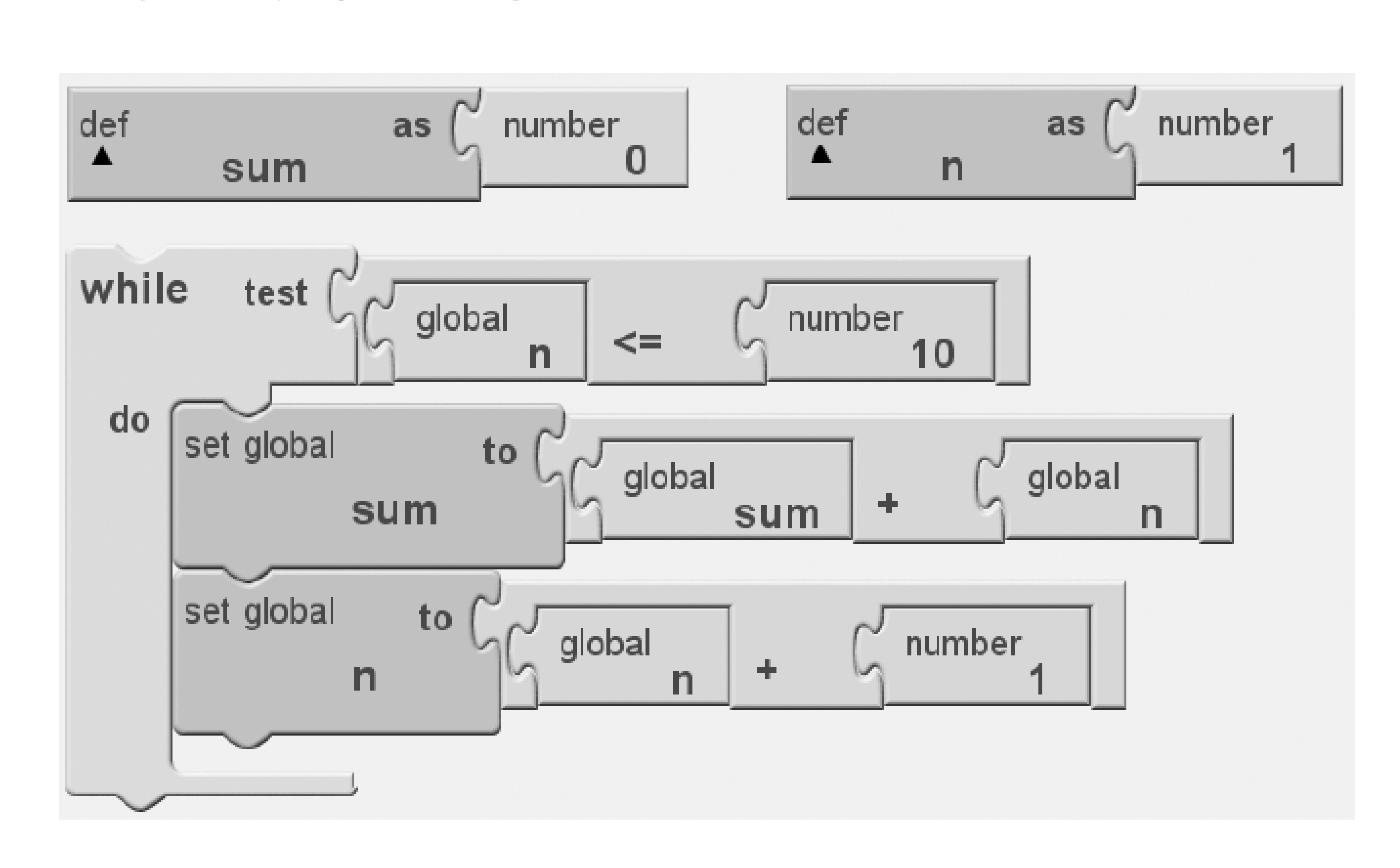


图 4.45　while 模块的累加数字示例

4.5 函　　数

有一个经典的脑筋急转弯,可以用来说明什么是函数。还记得把大象放到冰箱里面要几个步骤吗？第一步是打开冰箱门,第二步是把大象放进冰箱,第三部是关上冰箱门。

那么,把李雷放到冰箱里面的过程是什么？仍然是三步：第一步是打开冰箱门,第二步是把李雷放进冰箱,第三步是关上冰箱门。那如果要把韩梅梅放到冰箱里面的步骤,读者也很容易就会想到了。

如果用函数来分析上面的问题,可以把“放进冰箱”作为一个函数,把上述的三个步骤作为函数的内容,而到底把谁放进冰箱就成了函数的参数。

定义函数的好处就是可以避免重复工作。举个例子说明这个好处,如果要把李雷、韩梅梅、林涛和露西都放到冰箱里面,那么就会出现如下的过程：第一步是打开冰箱门,第二步是把李雷放进冰箱,第三步是关上冰箱门,第四步是打开冰箱门,第五步是把韩梅梅放进冰箱,第六步是关上冰箱门,第七步是打开冰箱门,第八步是把林涛放进冰箱……一共要经历 12 个步骤才可以完成。

下面用函数来简化过程。第一步是定义一个“放进冰箱”函数,第二步是调用函数“放进冰箱”将李雷放进冰箱,第三步是调用函数将韩梅梅放进冰箱,第四步是调用函数将林涛放进冰箱,第五步是调用函数将露西放进冰箱。使用函数后,经历的步骤降到了 5 个。

这样看来,函数将重复的动作进行了统一的描述,不同的地方提出来作为参数,以后再出现这样的动作,只要使用不同参数调用函数就可以实现重复的工作了。

App Inventor 提供了自定义函数功能,允许用户将实现一定功能的模块进行封装,并为其封装后的模块进行命名,这样就创建了一个函数,并可以被其他模块调用,实现了基本的代码复用。代码复用不仅降低了程序的错误率,而且提高了程序的可维护性。

4.5.1 定义与调用

在 App Inventor 中,定义函数的过程与定义变量的过程十分相似。首先在 Built-In→Definition 中找到函数定义模块 procedure 和 procedureWithResult,如图 4.46 所示。

App Inventor 中有两个函数定义模块,区别在于 procedure 模块没有返回值,而 procedureWithResult 有返回值。两个函数定义模块结构如图 4.47 所示。

下面以 procedure 模块为例,说明如何定义函数。定义过程分为两步,第一步是给函数命名,第二步是实现函数功能。

第一步,在路径 Built-In→Definition→procedure 中选取 procedure 模块,将模块名 procedure 更改为自定义名称 SumNum,如图 4.48 所示。

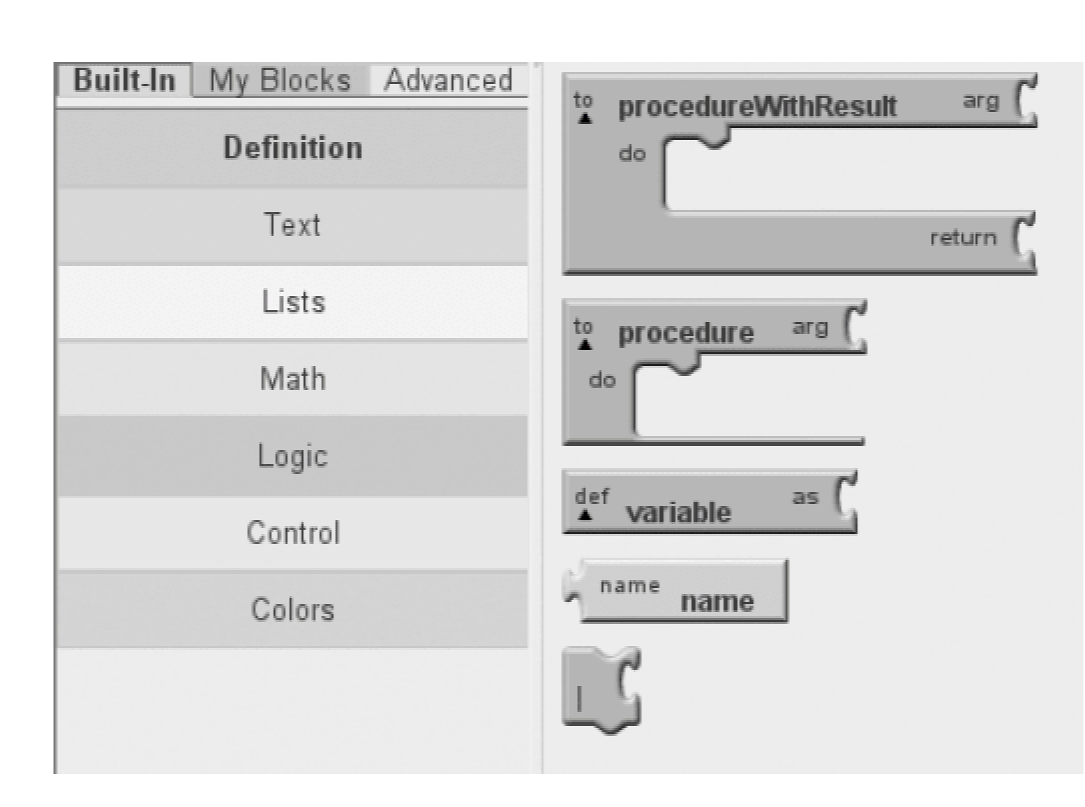


图 4.46　模块编辑器中的函数定义模块的位置

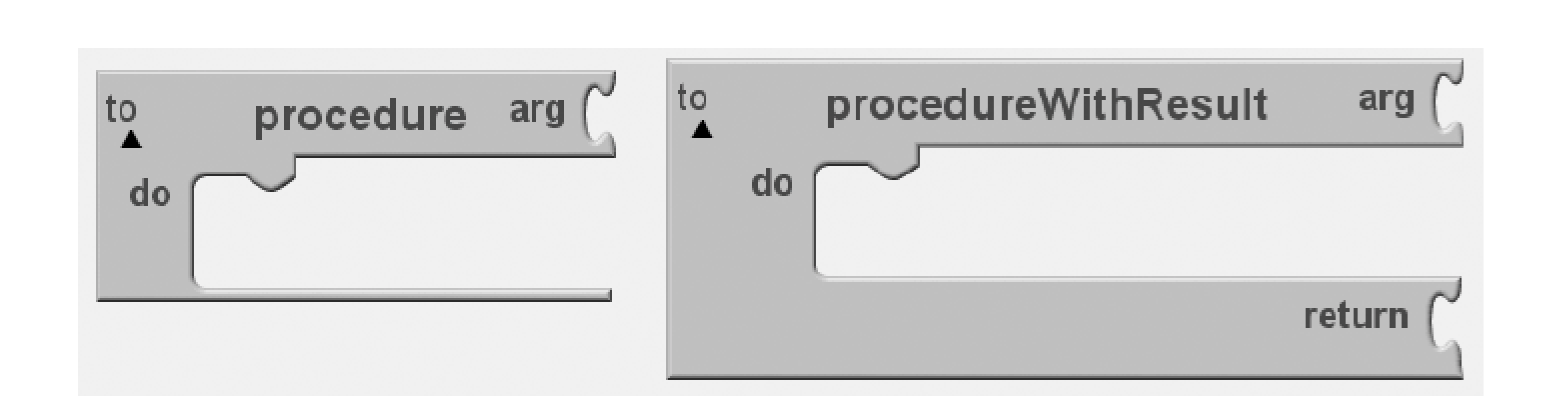


图 4.47　两个函数定义模块

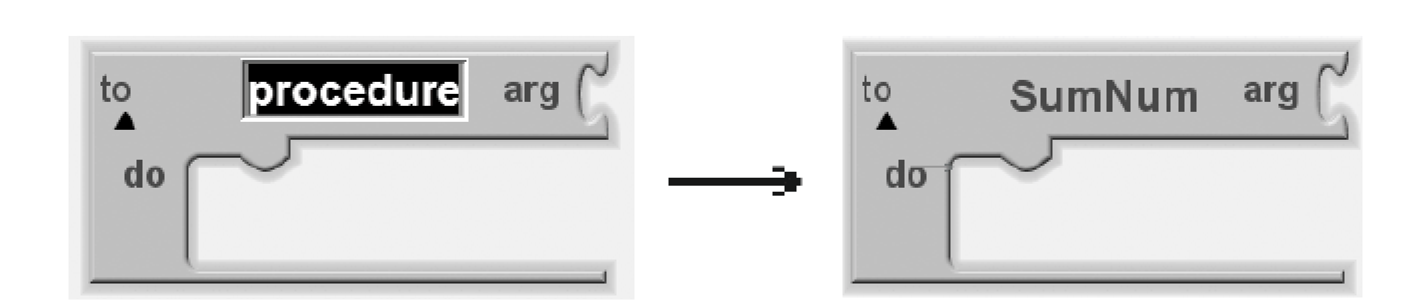


图 4.48　定义函数名称

第二步，实现函数功能，这里使用了 while 模块中的"数字累加"示例，将其封装在函数 SumNum 中，如图 4.49 所示。为了实现累加功能，在函数模块外定义了两个全局变量 sum 和 n。

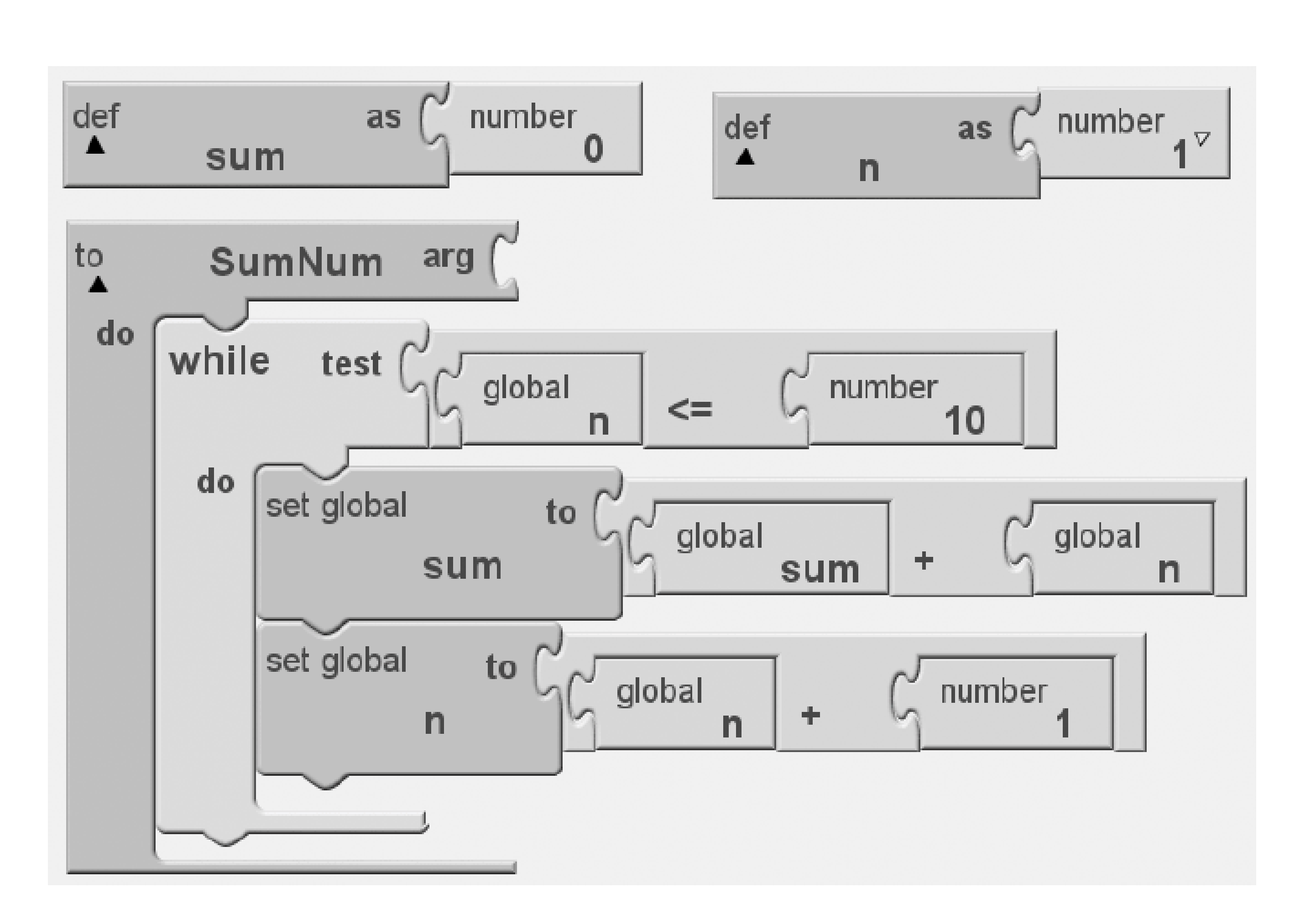


图 4.49　SumNum 函数

函数定义完成后，怎样才能发挥它的作用呢？定义完成后的函数只是一个单独的功能模块，只有在程序执行的过程中调用了这个函数，该函数才能被执行而发挥自己的功能。那么函数是怎样被程序调用的呢？

在模块编辑器的 My Blocks 中的第一个选项块就是 My Definitions，顾名思义，这里面包含所有自定义的变量和函数的调用模块。当一个函数定义完成后，在 My Definitions 中就会出现该函数的调用模块。

对于上面定义的函数 SumNum 而言，在 My Definitions 中则会包含该函数的调用模块，如图 4.50 所示。

在程序执行到某一步时，若要开始调用某个函数，则只需在程序的这个地方插入函数调用模块，当程序执行函数调用模块时，则函数被调用，开始执行函数体中的内容。

在图 4.51 中，在用户单击 Button1 按钮后，程序调用 SumNum 函数，然后进入 SumNum 函数体中，在执行完 SumNum 函数体中所有的模块后，跳回到调用 SumNum

函数的位置,继续执行下一个修改 Button1 文字内容的模块。

call
SumNum

图 4.50 SumNum 函数的调用模块

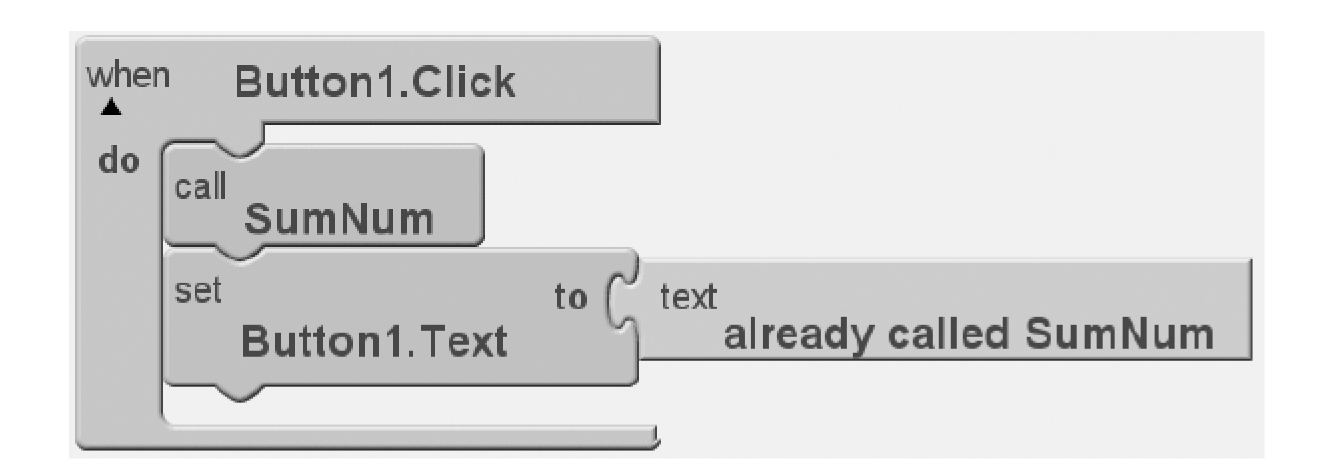


图 4.51 调用 SumNum 函数

4.5.2 函数参数

像大部分模块一样,函数模块也有自己的参数,可以根据参数的不同,产生不同的调用结果。创建函数的目的是消除代码的冗余,提高代码的效率,最重要的是函数要尽量做到通用性好,这就需要在函数中定义参数,使函数具有广泛性和普遍性。

下面继续修改"数字累加"示例,使函数可以自定义循环的次数。在未修改前,函数内部的 while 循环次数固定在 10 次,且不能在函数被调用时修改。依照图 4.52 对函数做出修改,在槽 arg 上增加一个参数 N。参数模块在 Built-In→Definition→name,将名称从 name 修改为 N 即可。不要忘记在 while 的条件中,将小于等于 10,更改为小于等于 N。

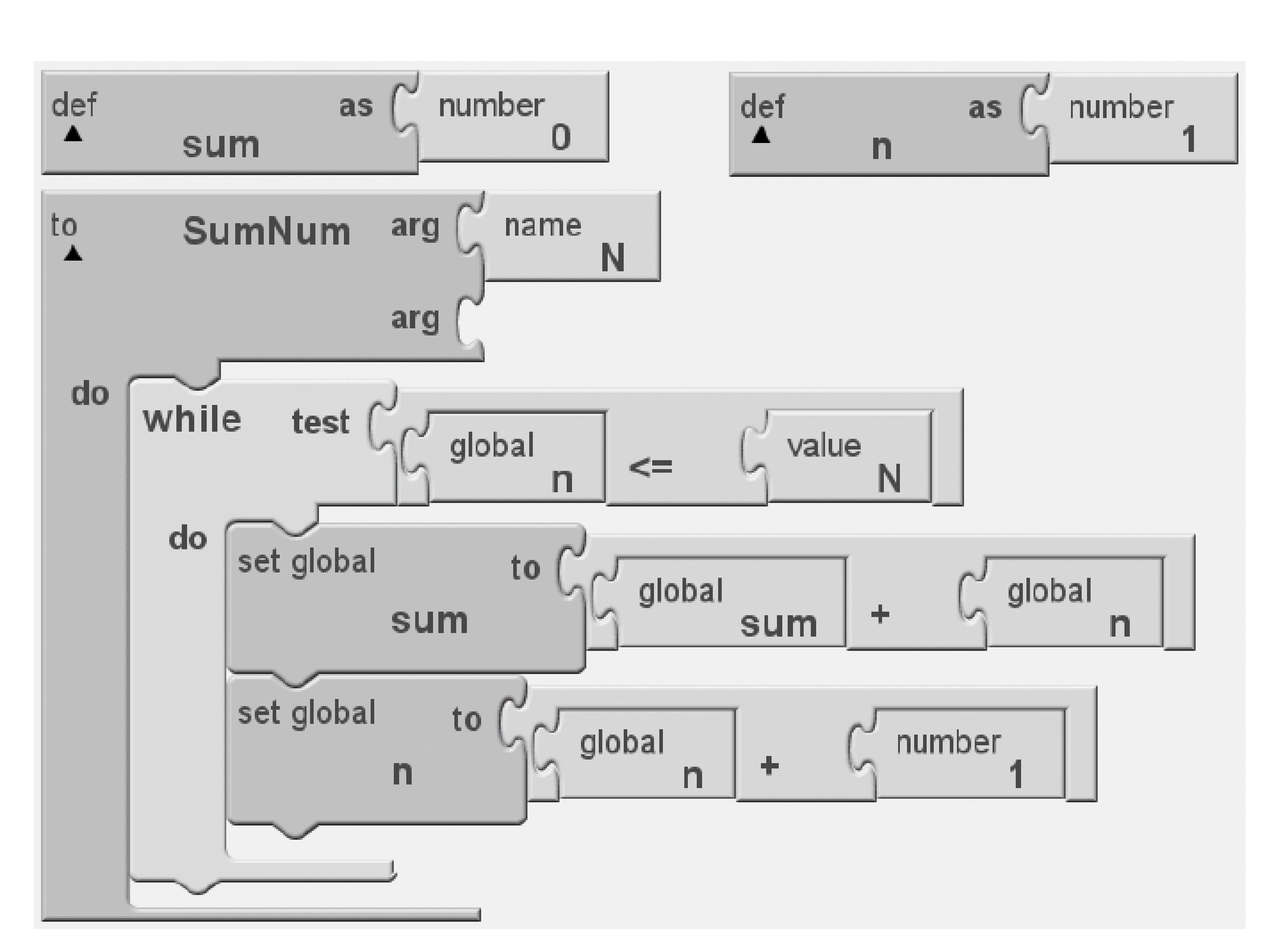


图 4.52 支持参数的 SumNum 函数

对上面函数的参数做出修改后,函数可以根据调用时的参数值决定函数的循环次数。例如在图 4.53 中,函数 SumNum 的调用模块多了一个槽 N,将数字 50 拼接在槽 N 上,就可以控制函数 SumNum 内部循环 50 次。

通过上面的内容不难发现,对函数参数的修改可以增加函数的通用性,函数参数的设定对于函数的作用范围和适用性具有较大的影响,所以在函数的创建过程中,要注意函数中参数的设置问题。

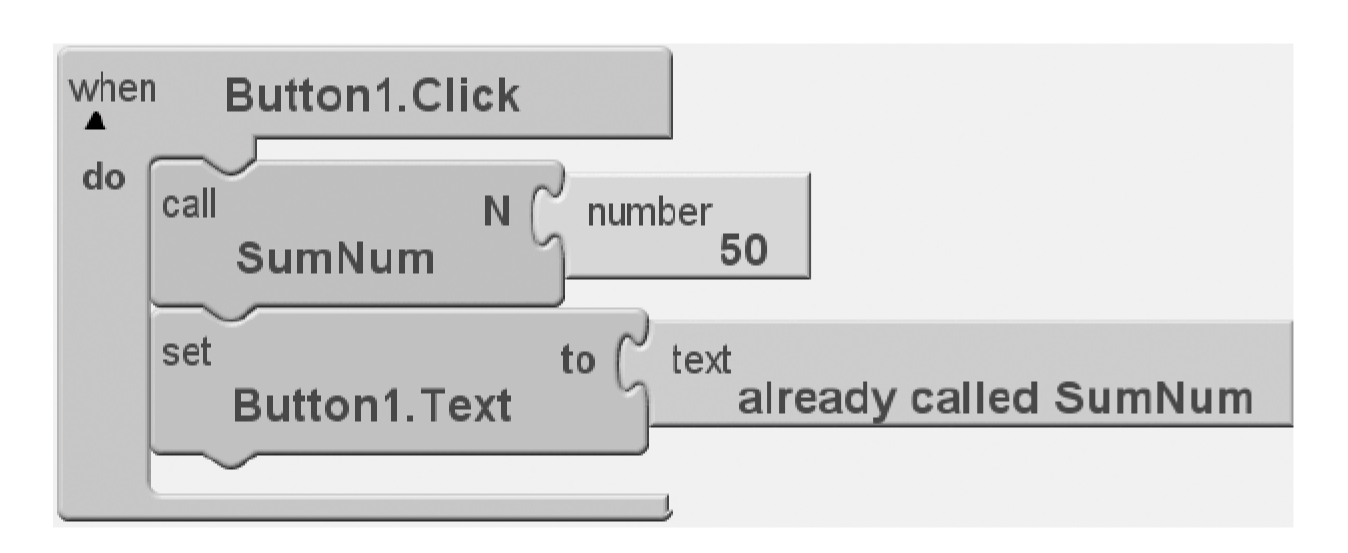


图 4.53 调用 SumNum 函数(参数 N 为 50)

4.5.3 函数返回值

函数的返回值可以直接供函数的调用者使用,从而简化代码逻辑。procedure 模块定义的函数则没有返回值,而 procedureWithResult 模块所定义的函数支持返回值。

procedureWithResult 模块的结构如图 4.54 所示,包含两个槽,槽 arg 是参数接口,与前面的 procedure 模块中的参数接口作用相同。另外一个槽 return 则是用来返回函数结果的接口,并且将函数返回值传给函数调用模块。

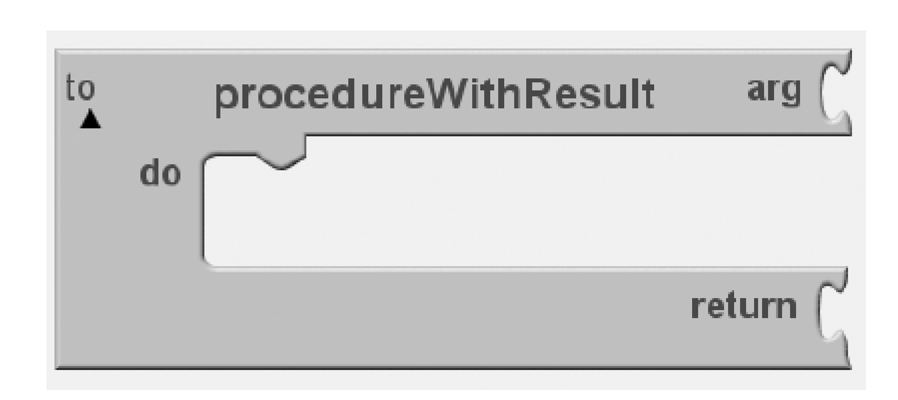


图 4.54 procedureWithResult 模块结构

下面使用 procedureWithResult 模块改造"数字累加"示例,为了与 SumNum 函数有所区别,该函数命名为 SumNumResult,如图 4.55 所示。SumNumResult 函数具有一个参数 N,以及一个返回值,将全局变量 sum 拼接在槽 return 上,函数会将累加值 sum 返回给函数的调用者。

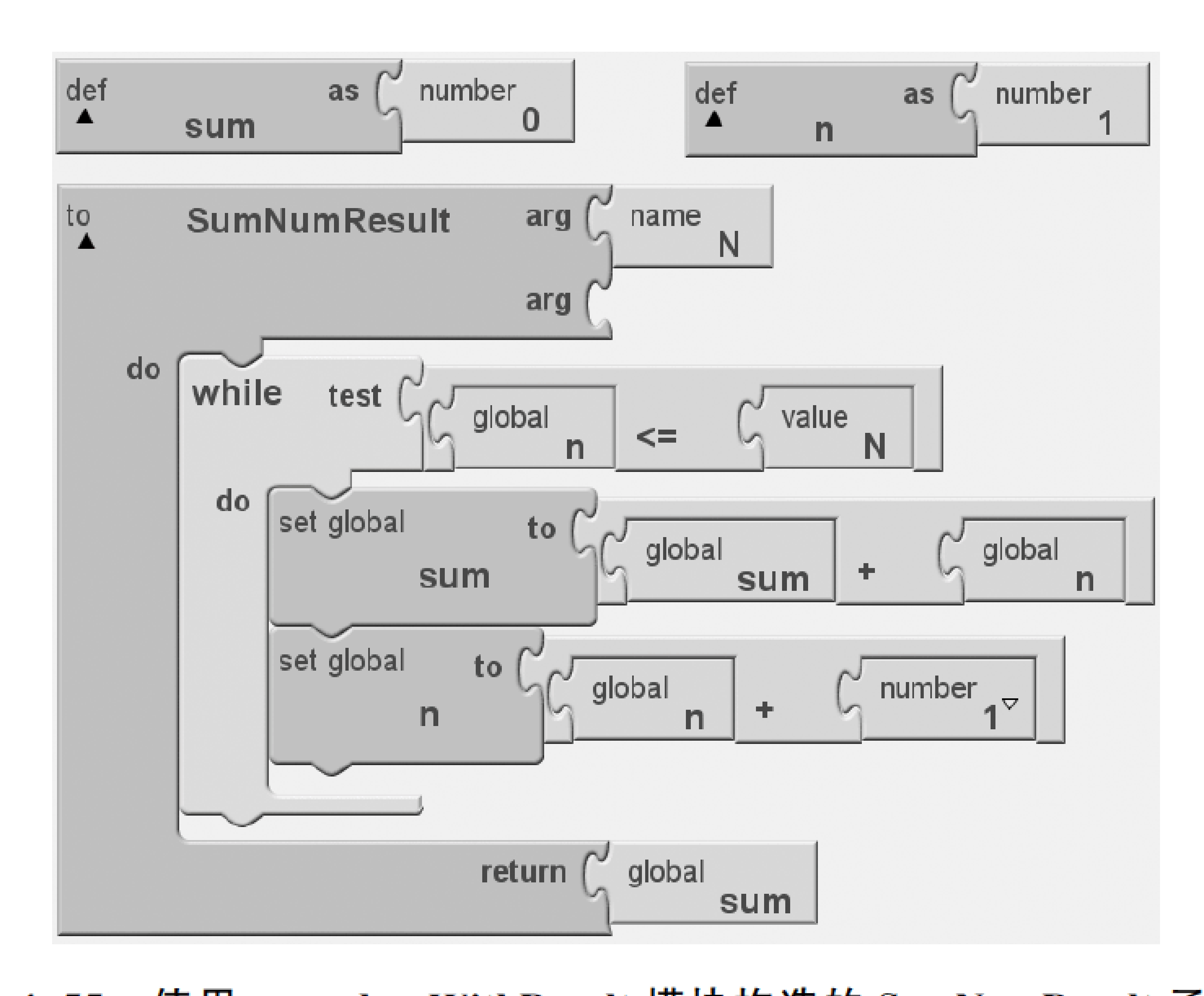


图 4.55 使用 procedureWithResult 模块构造的 SumNumResult 函数

因为具有返回值,函数 SumNumResult 的调用模块也有所变化,图 4.56 是函数 SumNumResult 和 SumNum 调用模块的对比,主要变化是槽的位置不同。

在按钮事件单击函数中,调用 SumNumResult 函数,其返回值可以直接给按钮的文



图 4.56 SumNumResult 和 SumNum 调用模块的对比

字属性赋值,如图 4.57 所示。

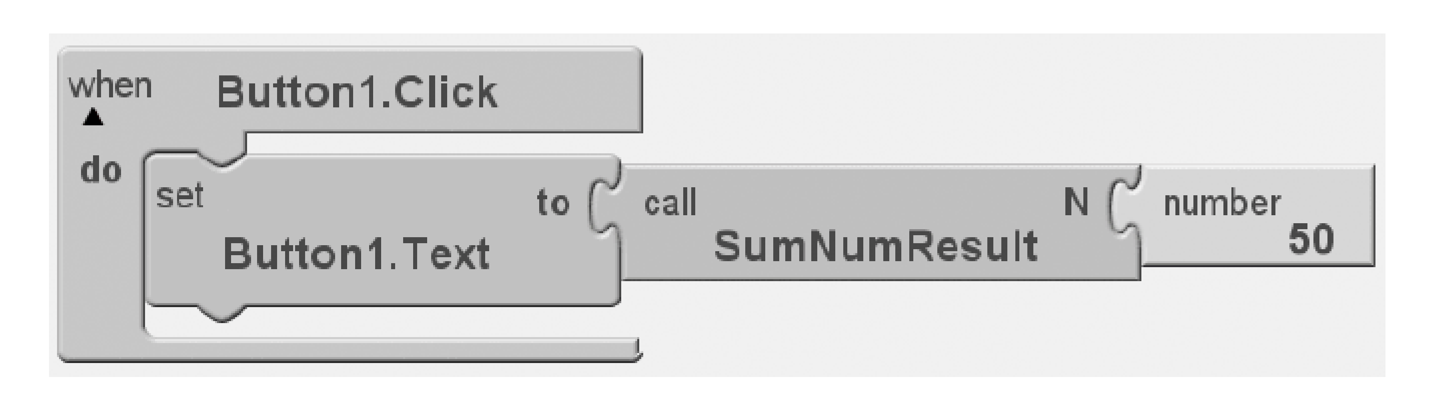


图 4.57 调用 SumNumResult 函数

4.6 模块分类

本章前面介绍的内容基本都是关于模块编辑器中 Built-In 区域的模块,但在前面的示例中也有一些与界面控件相关的模块,这些模块都在模块编辑器中的 My Blocks 中,并且根据控件的不同,它们各自所包含的功能模块也不尽相同。

模块编辑器的 My Blocks 区域中,包含的项有 My Definitions 项和界面控件项(例如 Button1、Label1 等)。开发人员在界面设计时每添加一个控件,那么在 My Blocks 中就会出现该控件项,单击控件项就会看到控件的相关功能模块。这些模块根据不同的性质和作用被分成三类,事件模块、属性模块和方法模块,并且为了便于区分,将这三类模块使用不同的颜色进行标识,其中事件模块为绿色,属性模块为蓝色,方法模块为紫色。

例如,在界面设计器中选定一个按钮控件 Button1,一个标签控件 Label1,并且界面只有一个屏幕 Screen1,这时候在模块编辑器的 My Blocks 中就会显示如图 4.58 所示的控件项。

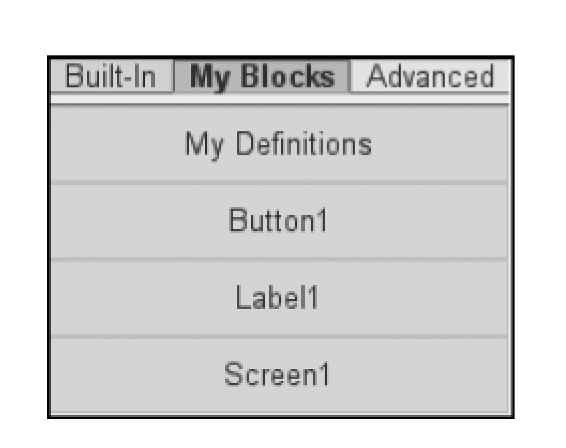


图 4.58 My Blocks 中的控件项

选择 Button1 和 Screen1 控件项后,其右侧将出现该控件的功能模块列表,如图 4.59 所示。在 Button1 的功能模块列表中,Button1.Click 模块是事件模块,所有的事件模块都会有"when do"的字样。set Button1.BackgroundColor to 是属性设置模块,所有的属性设置模块都有"set to"的字样;Button1.BackgroundColor 是属性值模块,这里将属性设置模块和属性值模块统称为属性模块。

在 Screen1 的功能模块列表中,Screen1.CloseScreenAnimation 是方法模块,所有的方法模块都有"call"的字样。

知道了如何区分事件、属性和方法模块后,下面对 My Blocks 中的 My Definitions 区域做简单的介绍。

My Definitions 是自定义函数、全局变量和参数的集合,如图 4.60 所示。上方第一

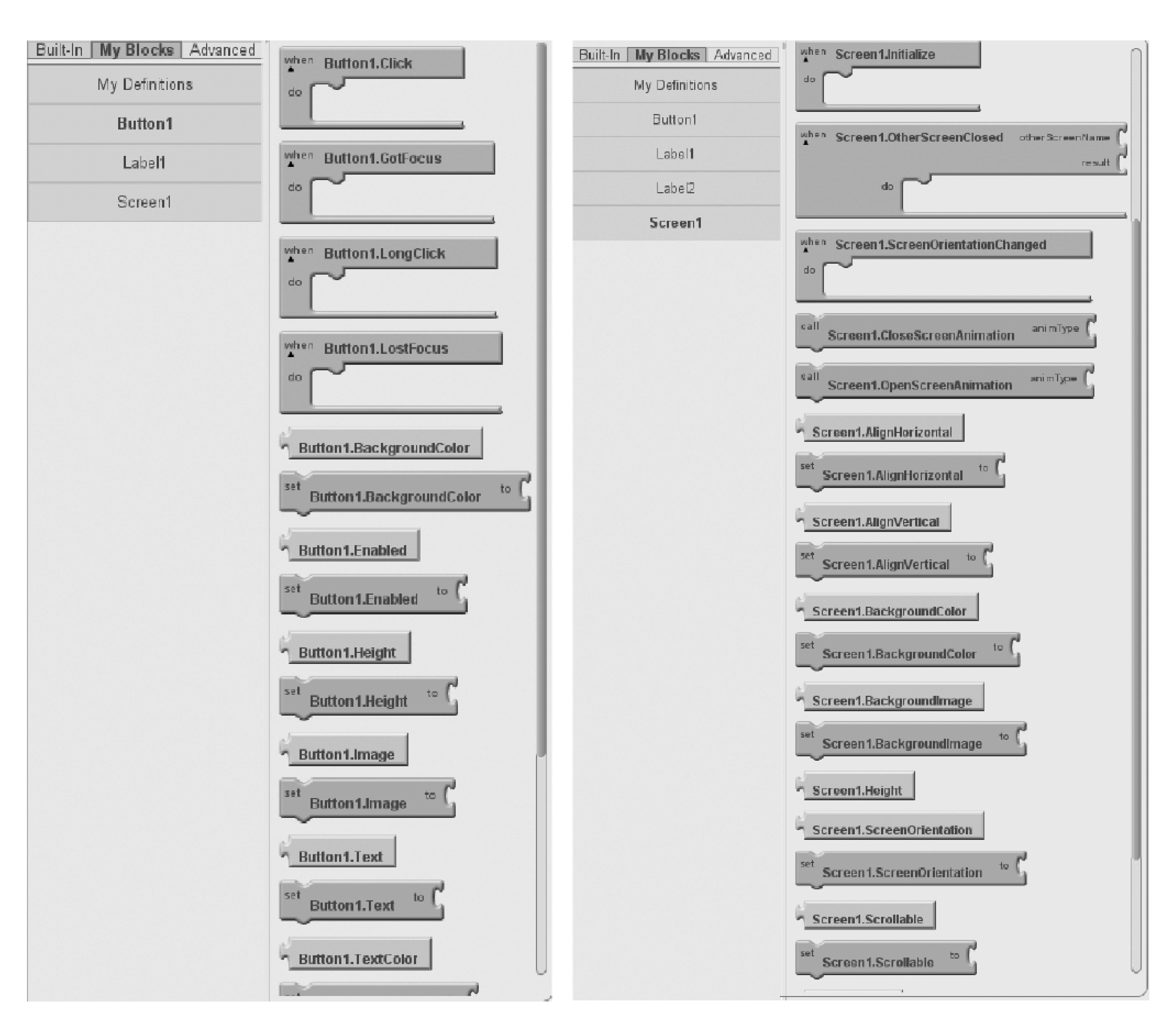


图 4.59　Button1 和 Screen1 的功能模块列表

个模块是全局变量 n，标识是“global”，第二个就是全局变量 n 的赋值模块。上方第三个模块是参数 N，在函数中被定义和使用。最后两个模块是函数 SumNum 和函数 SumNumResult 的调用模块。

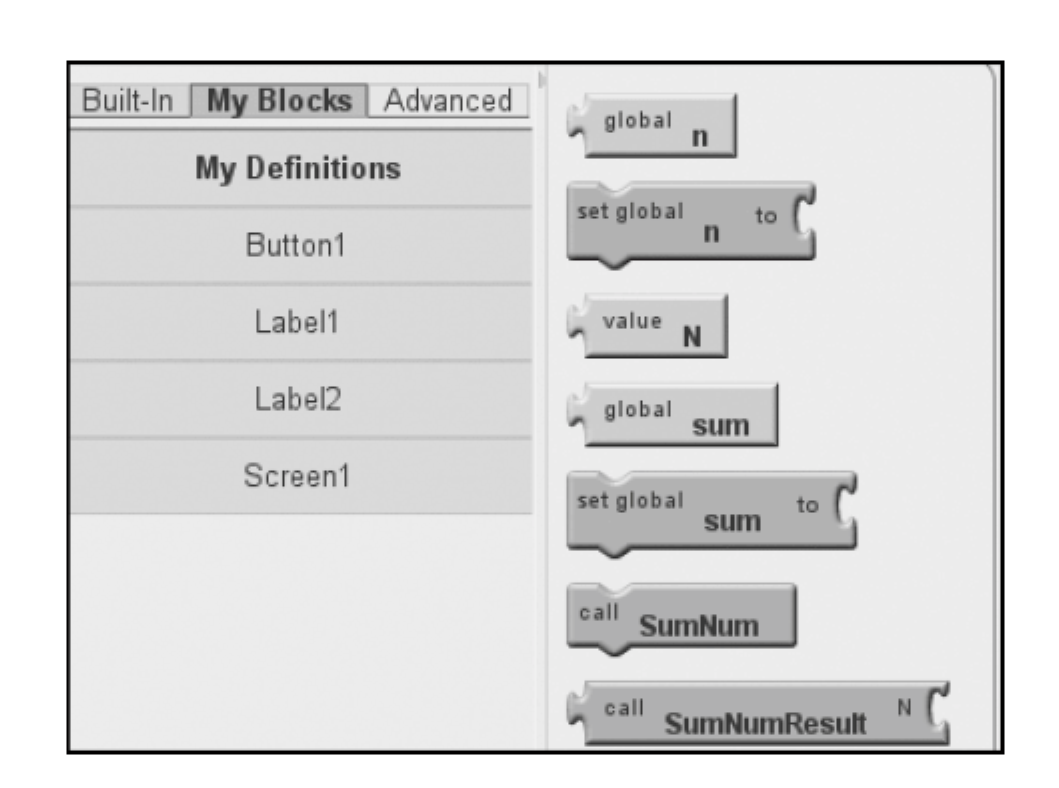


图 4.60　My Definitions 是自定义函数、全局变量和参数的集合

4.6.1　事件

事件是指某种特定的情况发生，例如单击按钮、获取焦点等。在 App Inventor 中事件是开启程序执行的关键，通过事件的触发，才能执行事件处理模块，进而运行程序中所包含的功能模块。

例如，通过对单击 Button1，触发按钮单击事件(Button1. Click)，则程序可以将标签 Label1 的背景颜色设置为红色，程序结构如图 4.61 所示。

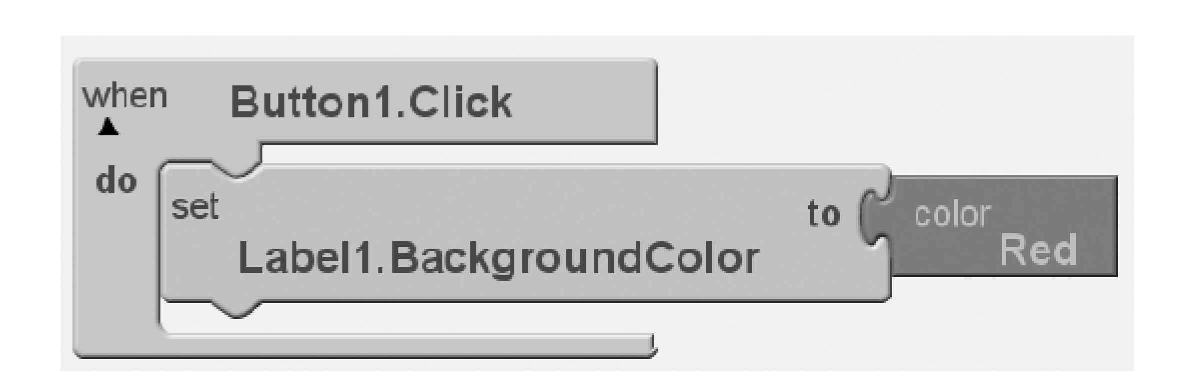


图 4.61 Button1 单击事件触发程序的执行

4.6.2 属性

属性是指控件所具有的性质，如宽度、高度、背景颜色和本文内容等。在 App Inventor 的模块编辑器中，为每个控件的不同属性都提供了相应的属性模块，并且提供了与之对应的属性设置模块。在程序设计过程中，可以通过属性设置模块对控件的属性值进行修改。

在图 4.62 的例子中，set Label2. BackgroundColor to 是属性设置模块，而 Label1. BackgroundColor 是属性值模块。通过将 set Label2. BackgroundColor to 模块和 Label1. BackgroundColor 模块的拼接，将标签 Label1 的背景颜色赋值给了标签 Label2。

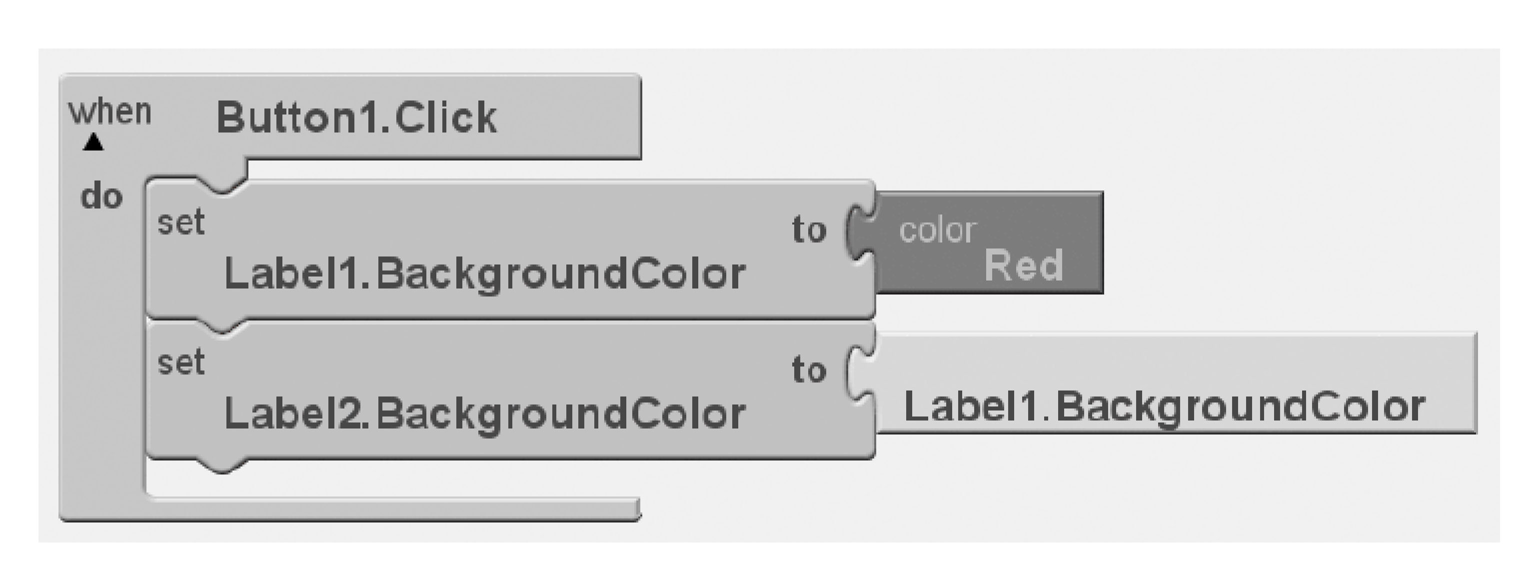


图 4.62 对 Label1 和 Label2 的背景颜色属性设置

4.6.3 方法

在 App Inventor 的控件中，很多控件都有自己的方法，例如时钟控件的增加秒数方法(Clock1. AddSeconds)、短消息控件的发送消息的方法(Texting1. SendMessage)等，如图 4.63 所示。

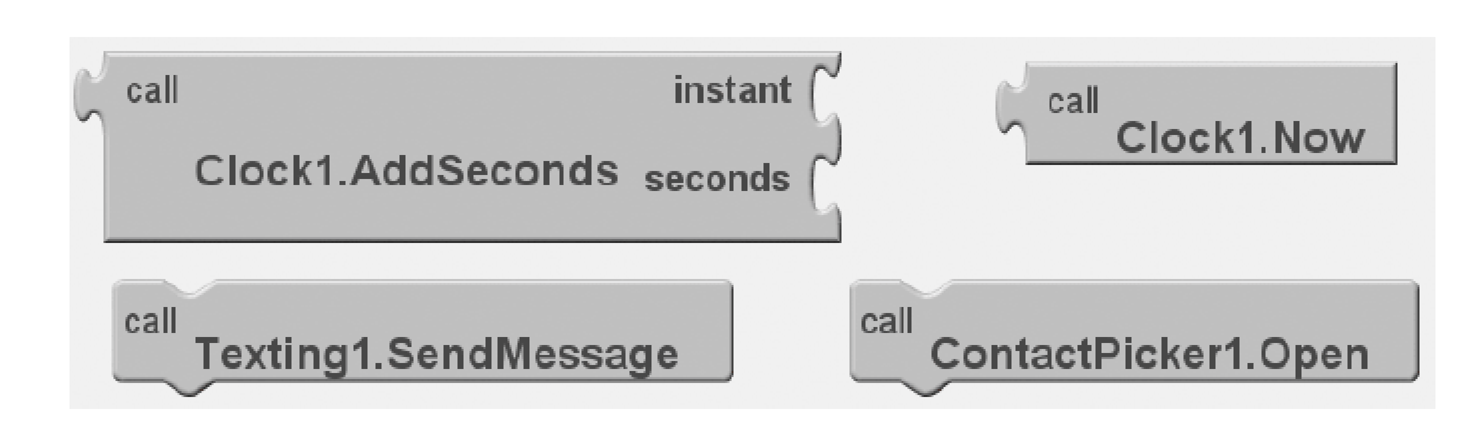


图 4.63 控件的方法模块

用户自定义的函数，也可以是一种方法。沿用上面的示例，创建一个函数 setColor，该函数体中修改两个标签的背景颜色，然后在 Button1 的单击事件中，调用自定义的函

数，如图 4.64 所示。

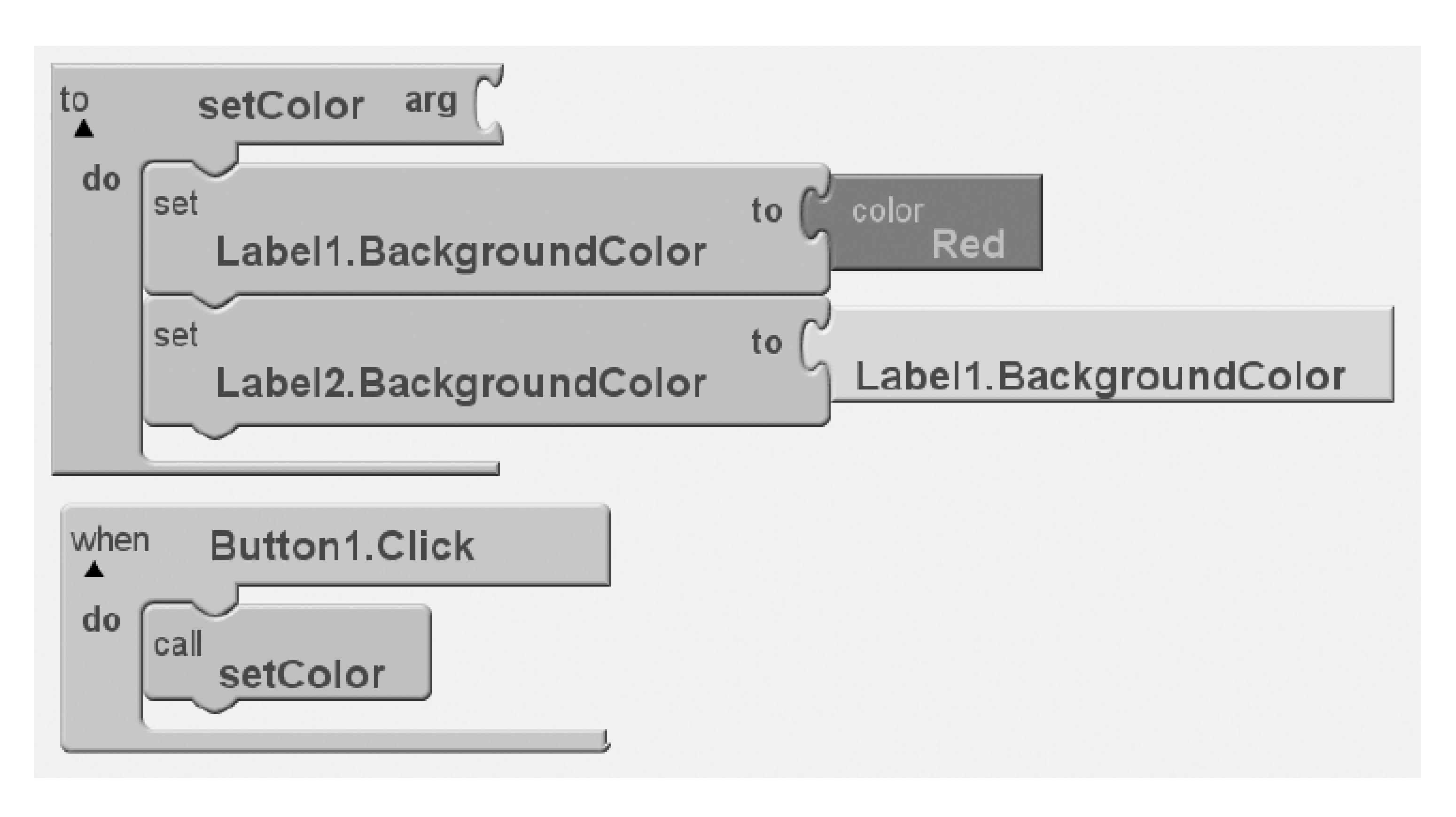


图 4.64 函数方法调用

习 题

1. 简述程序设计中的几种分支和循环模块结构的特点。
2. 简述静态列表和动态列表的区别。
3. 设计一个可以计算三个整数平均值的函数。
4. 简述事件、属性和方法的区别。

第 5 章



用户界面设计

用户界面是应用程序开发的重要组成部分,决定了应用程序是否美观、易用。App Inventor 使用可视化的、所见即所得的方式来搭建应用程序的界面部分。通过本章的学习,读者将熟悉界面设计中常用的基本控件和布局格式。

本章学习目标:

- 了解控件分类和基本用途;
- 掌握基础控件的使用方法;
- 掌握界面布局的使用方法;
- 掌握媒体控件的使用方法;
- 了解社交控件的使用方法。

5.1 控件概述

App Inventor 的界面开发采用可视化的方法,通过将界面控件拖曳到界面编辑器内,在设计区中直接生成界面效果图,这些可以被拖曳的界面元素就是控件。

App Inventor 的控件库中有丰富的界面控件。控件库细分为 9 个子类,共 51 个控件,控件的类别和每类控件的数量如表 5.1 所示。

表 5.1 界面控件类别

类　别	说　明	类　别	说　明
Basic	基础控件,10 个	Screen Arrangement	屏幕布局控件,3 个
Media	媒体控件,6 个	Other stuff	其他控件,9 个
Animation	动画控件,2 个	Not ready for prime time	不成熟控件,5 个
Social	社交控件,6 个	LEGO MINIDSTORMS	乐高机器人控件,7 个
Sensors	传感器控件,3 个		

基础控件(Basic)中存储着界面开发时使用率最高的控件,如按钮、标签和复选框等控件。媒体控件(Media)中主要是用来播放声音、视频,以及进行录像的控件。动画控件(Animation)是用来开发游戏的移动效果的控件。传感器控件(Sensors)包含最常用的加

速度传感器、位置传感器和方向传感器控件。屏幕布局控件(Screen Arrangement)是用于设定屏幕中元素排列方式的控件。其他控件(Other stuff)中包含没有被划归到其他类别的控件,如蓝牙客户服务端、语音识别、小型 Web 数据库等控件。不成熟控件(Not ready for prime time)是在试用期的一些控件,这些控件未来可能会进入其他控件类中,成为正式控件,也有可能从控件库中被删除。乐高机器人控件(LEGO MINIDSTORMS)是乐高 NXT 智慧型机器人的开发控件。

5.2 基础控件

基础控件类中的控件具有较高的使用频率,这些控件包括按钮、画布、复选框、图片、标签、下拉菜单、密码框、文本框和时钟。微型数据库(TinyDB)控件的使用方法将在第7章中进行介绍。

5.2.1 按钮、标签和图像

按钮(Button)是界面上最基本的组件,主要提供单击式的触发操作。通过设置按钮的事件和状态,可以实现基本的人机交互功能。在基础控件类中的第一项就是按钮,如图 5.1 所示,可见按钮在界面开发中的重要程度。所有控件的右侧都有一个问号,单击问号可以获取到控件的使用说明。

App Inventor 的按钮是可以添加图片的,相当于传统意义的“图像按钮(ImageButton)”,而且可以设置按钮自身的形状,例如设置成长方形、椭圆形的按钮。这些改变只要简单地修改按钮属性就可以实现。图 5.2 给出了按钮的属性编辑栏,在这里可以方便地修改按钮属性。

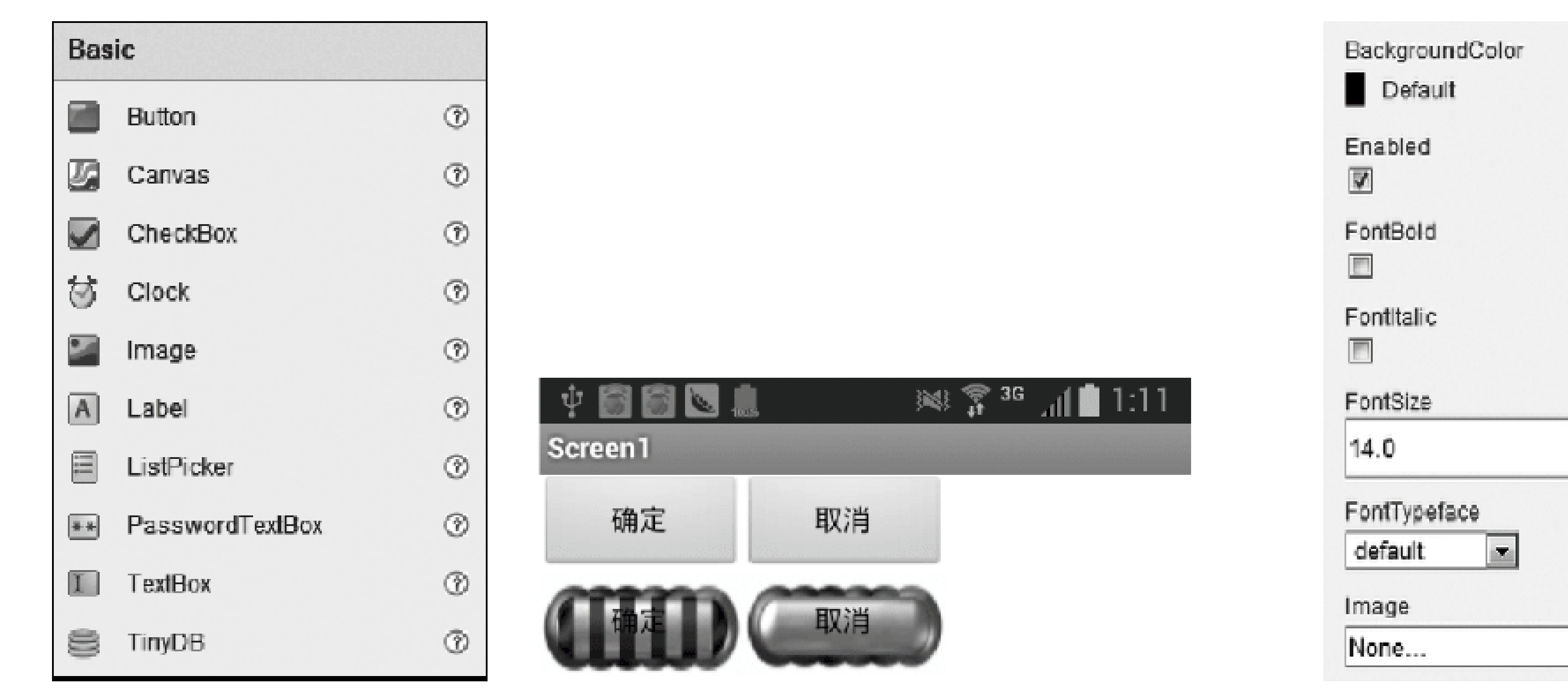


图 5.1 基础控件类和按钮控件

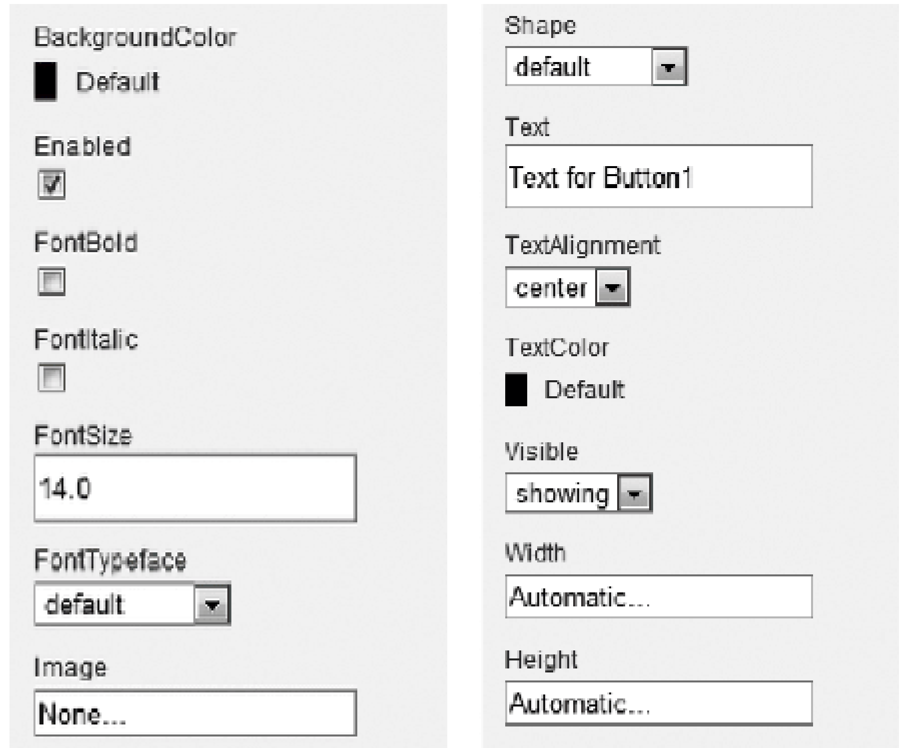


图 5.2 按钮控件的属性编辑栏

在后面介绍其他控件时,如 Enabled、TextAlignment、Width 和 Height 属性会频繁地出现,在这里先对这些通用属性进行介绍。

Enabled 属性标识着控件是否可用。将控件的 Enabled 属性设置为不可用,在界面上的控件会变为灰色,且不接受任何用户操作。在图 5.3 中,Button1 的 Enabled 属性为可用,而 Button2 的 Enabled 属性为不可用。

TextAlignment 属性是文字的对齐方式，支持左对齐、右对齐和居中对齐三种对齐方式，图 5.4 中的三个按钮从上到下分别是左对齐、居中对齐和右对齐的显示文字。



图 5.3 Enabled 属性　　**图 5.4 TextAlignment 属性**

Width(宽度)和 Height(高度)属性也是每个控件必有的属性，控制着控件在界面上显示的宽度和高度，支持自动(Automatic)、填充(Fill Parent)和固定尺寸(Pixels)三种选择。

如表 5.2 所示给出了按钮所有支持的属性，以及属性说明。

表 5.2 按钮控件的属性

属　性	说　明
Background Color	设置按钮的背景色
Enabled	设置按钮是否可用
FontBold	设置字体加粗
FontItalic	设置字体倾斜
FontSize	设置字体大小
FontTypeface	设置字体类型
Image	设置按钮的背景图案
Shape	设置按钮的形状，如圆角按钮、矩形按钮等
Text	设置按钮上显示的文字，如果清空则不在按钮上显示任何文字
TextAlignment	设置按钮上文字的对齐方式
TextColor	设置文本的颜色
Visible	设置按钮是否可见
Width	设置按钮的宽度
Height	设置按钮的高度

Shape 属性可以控制按钮的外部形状，目前支持圆角形、矩形和椭圆形的按钮，如图 5.5 所示。

按钮支持的事件有单击事件(Click)、长时间单击事件(LongClick)、获取焦点事件(GetFocus)和失去焦点事件(LostFocus)，如图 5.6 所示。单击事件是将手指按在按钮上，然后抬起时产生。手指按下后立即抬起才会产生单击事件，否则会产生长时间单击事件。如果长时间按在按钮上时，即使没有抬起手指，也会引发长时间单击事件。

Text for Button1
Text for Button2
Text for Button3

图 5.5 Shape 属性

标签(Label)主要起到文字显示的作用，但标签不允许用户进行输入操作，只能够显示文字信息，如图 5.7 所示。标签只有属性，没有事件，因此标签不支持单击、长时间单击或焦点切换等操作。

when Button1.Click
do
when Button1.GotFocus
do
when Button1.LongClick
do
when Button1.LostFocus
do

图 5.6　按钮控件事件

这是默认标签
这是带背景颜色的标签
这是修改过字体属性的标签

图 5.7　标签控件

标签与按钮有部分属性是相同的，因此不再重复介绍，标签支持的属性可以参考表 5.3。

表 5.3　标签控件的属性

属　　性	说　　明	属　　性	说　　明
Background Color	设置标签背景色	TextAlignment	设置标签内文字的对齐方式
FontBold	设置字体加粗	TextColor	设置文本的颜色
FontItalic	设置字体倾斜	Visible	设置标签是否可见
FontSize	设置字体大小	Width	设置标签的宽度
FontTypeface	设置字体类型	Height	设置标签的高度
Text	设置标签栏里显示的文字		

图像(Image)控件用于在界面上显示各种图像文件，图像控件也不支持事件，所支持的属性如表 5.4 所示。图像控件虽然功能单一，但由于属性简单、易于使用，因此在应用开发过程中会经常被使用到。

表 5.4　图像控件的属性

属　　性	说　　明	属　　性	说　　明
Picture	选择显示的图像	Width	设置图像的宽度
Visible	设置图像是否可见	Height	设置图像的高度

图像控件的使用会极大地增强应用程序的美观程度，巧妙地使用图像控件是一项技巧，下面以 FourSeasons 示例进行讲解，介绍如何使用图像控件、按钮和标签控件，完成一个稍微复杂一些的应用程序。



图 5.8　FourSeasons 示例运行界面

FourSeasons 示例的运行界面图 5.8 所示，用户通过单击下方标有“春天”、“夏天”、“秋天”和“冬天”的 4 个按钮，切换界面上方的表示季节的图片。（注：FourSeasons 示例的“春”、“夏”、“秋”、“冬”的风景图片，素材来源于 http：// www. nipic. com/show/3/35/5067160k709b6c7f. html。）

FourSeasons 示例的界面简单明了，如图 5.9 所示，最上方的显示“四季更替”的是一个标签，标签的名

称为 Title。下方是显示季节的图像控件，名称为 picture。再下方是一个标签控件，名称为 text，用来显示用户单击按钮后的提示信息。最下方是 4 个按钮控件，分别命名为 Spring、Summer、Autumn 和 Winter。



图 5.9　FourSeasons 示例的界面设计图

在工程中使用图像控件，将要使用的图片通过 Upload new 按钮加载到工程中，Upload new 按钮在界面编辑器的资源区(Media)中，如图 5.10 所示。

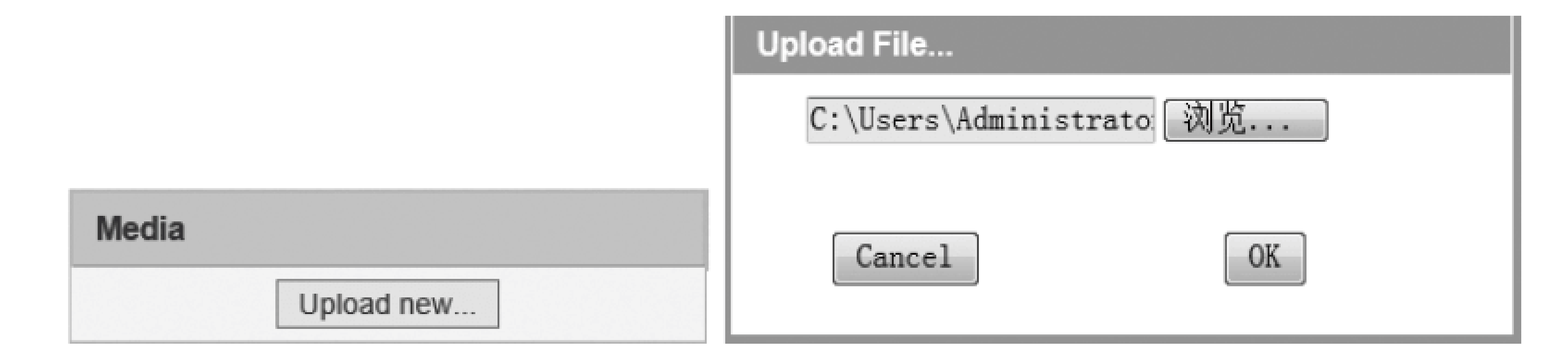


图 5.10　将图片加载到工程中

FourSeasons 示例的逻辑非常简单："单击不同的按钮，显示不同的图片和带不同颜色的文本信息"，这样需要为每个按钮添加一个单击事件，并在单击事件中修改显示的图片，以及显示不同颜色的文字。FourSeasons 示例的逻辑模块图如图 5.11 所示。

5.2.2　文本框、复选框和密码框

文本框(TextBox)是一种供用户进行输入文字的控件。虽然文本框可以显示文字信息，但其主要的功能还是为用户提供输入信息的区域，比如登录框、搜索栏或是编辑文字

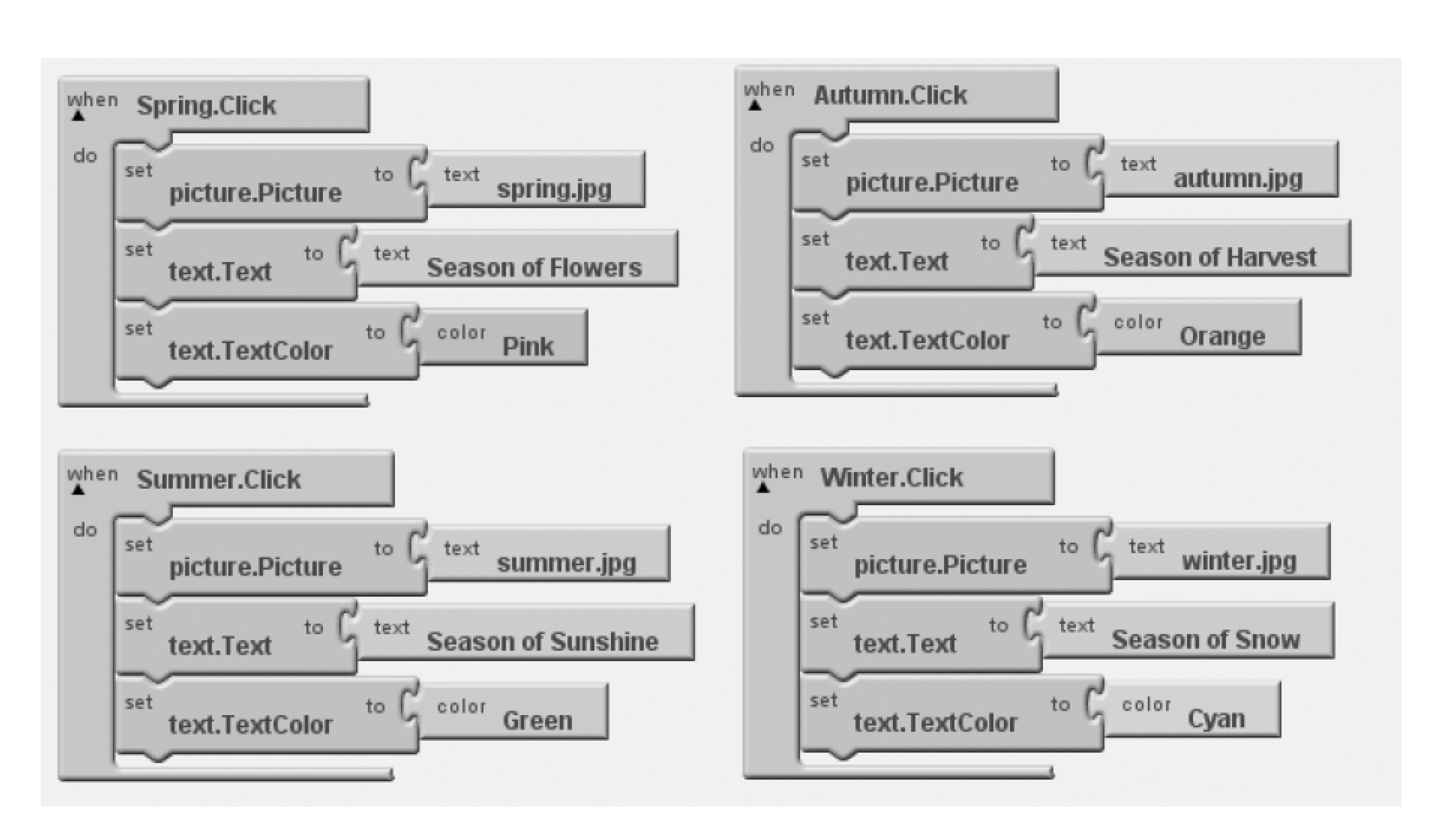


图 5.11　FourSeasons 示例的全部逻辑模块

的写字板等。文本框的使用范围非常广泛，是基本的信息输入控件之一，是很多应用程序中不可或缺的组成部分。

当文本框中的内容为空时，会自动在文本框中以灰色文字显示提示信息(Hint)。提示信息在文本框的 Hint 属性中设置，可以提示用户在文本框内输入信息的类型。例如，在图 5.12 中，上方的文本框中显示了“Hint for TextBox1”的提示信息。

Hint for TextBox1

这是一个可以显示多行文本的文本框，这是一个可以显示多行文本的文本框，这是一个可以显示多行文本的文本框，这是一个可以显示多行文本的文本框，这是一个可以显示多行文本的文本框

图 5.12　文本框

MultiLine 属性可以控制文本框显示单行文本或是多行文本，在图 5.12 中，上方的文本框的 MultiLine 属性没有被设置，无论用户输入多少信息，文本框会一直保持单行的状态；下方文本框的 MultiLine 属性已经被设置，因此在用户输入较多信息时，文本框会自动变为多行显示模式。

NumbersOnly 属性控制了文本框中可输入信息的类型，如果该属性被设置为 true，文本框只接受数字，不接受用户输入其他字符。

文本框控件的全部属性可参考如表 5.5。

表 5.5　文本框控件的属性

属　性	说　明
Background Color	设置文本框的背景色，默认为白色
Enabled	设置文本框是否可用
FontBold	设置字体加粗
FontItalic	设置字体倾斜
FontSize	设置字体大小
FontTypeface	设置字体类型
Hint	设置文本框的提示信息

续表

属性	说明
MultiLine	设置是否支持多行显示
NumbersOnly	设置是否只允许输入数字
Text	设置文本框上默认显示的文字内容
TextAlignment	设置按钮上文字的对齐方式
TextColor	设置文本的颜色
Visible	设置文本框是否可见
Width	设置文本框的宽度
Height	设置文本框的高度

文本框仅支持获取焦点事件(GotFocus)和失去焦点事件(LostFocus),如图 5.13 所示。

复选框(CheckBox)是可以同时选中多项的选项框,供用户在不同选项间进行多项选择时使用,在程序当中起到条件识别的作用,如图 5.14 所示。



图 5.13 文本框控件事件

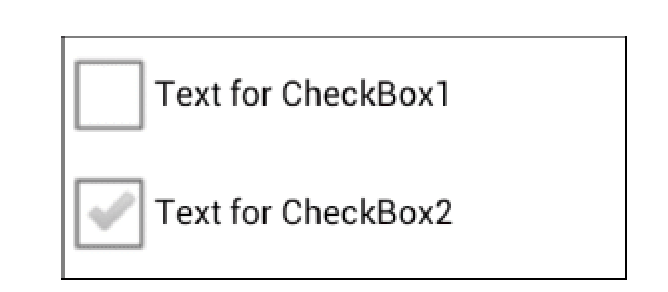


图 5.14 复选框

Checked 属性是复选框的标志性属性,表示复选框是否被选中。复选框的高度(Height)和宽度(Width)设置的是复选框文字所占用的空间。复选框一般多个联合起来使用,作为多个条件的组合判断依据,不过也可以单独使用。复选框全部属性如表 5.6 所示。

表 5.6 复选框控件的属性

属性	说明
Background Color	设置复选框的背景色
Checked	设置复选框默认状态是否被选中
Enabled	设置复选框是否可用
FontBold	设置复选框字体加粗
FontItalic	设置复选框字体倾斜
FontSize	设置复选框字体大小
FontTypeface	设置复选框字体类型
Text	设置复选框的文字注释
TextColor	设置文本的颜色

续表

属　　性	说　　明
Visible	设置复选框是否可见
Width	设置复选框的宽度
Height	设置复选框的高度

复选框除了支持获取焦点事件(GotFocus)和失去焦点事件(LostFocus)以外,还支持选项更改(changed)事件,更改事件在复选框选择状态发生改变时产生,如图 5.15 所示。

密码框(PasswordTextBox)是一种特殊的文本框,一般用于接受用户输入密码,在用户输入信息时,密码框中会将输入的内容进行屏蔽处理。

图 5.16 上方的密码框是在输入内容为空时显示提示信息,下方的密码框是用户输入了密码信息,屏蔽处理后,显示为一行星号。密码框控件的可编辑属性如表 5.7 所示。

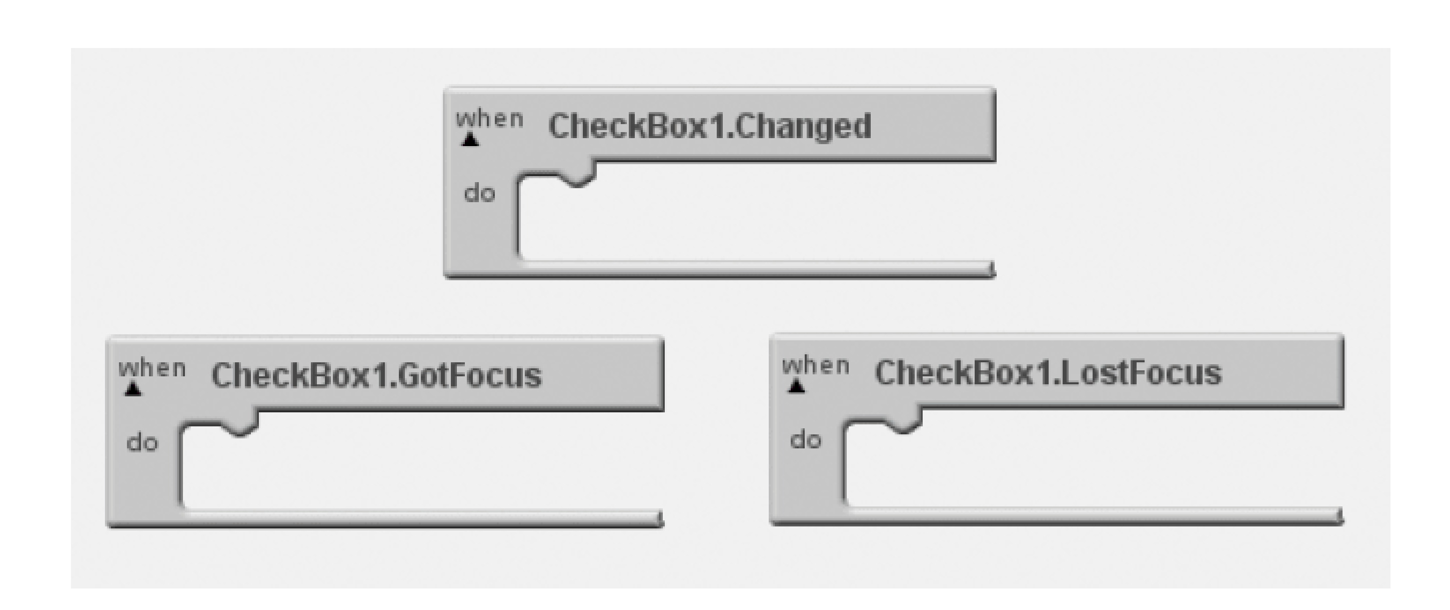


图 5.15　复选框控件事件



图 5.16　密码框

表 5.7　密码框控件属性

属　　性	说　　明
Background Color	设置密码框的背景色
Enabled	设置密码框是否可用
FontBold	设置密码框字体加粗
FontItalic	设置密码框字体倾斜
FontSize	设置密码框字体大小
FontTypeface	设置密码框字体类型
Hint	设置密码框的提示信息
Text	设置密码框里显示的文字
TextAlignment	设置密码框上文字的对齐方式
TextColor	设置文本的颜色
Visible	设置密码框是否可见
Width	设置密码框的宽度
Height	设置密码框的高度

密码框也仅支持获取焦点事件(GotFocus)和失去焦点事件(LostFocus),如图5.17所示。

密码框、文本框和复选框是制作登录界面的必备元素,下面以Login示例为例,介绍如何制作登录界面。

Login示例的运行界面如图5.18所示,用户在“输入用户名”文本框中输入用户名,在“输入密码”密码框中输入密码,单击“登录”按钮就可以完成登录过程。复选框用来控制是否显示提示信息。Login示例中只有一组正确的用户名和密码,用户名是user,密码是123456。如果用户输入正确,并成功登录,屏幕上方的文本框将显示登录成功的提示信息“login success”,如果没有正确地输入用户名和密码,屏幕上方的文本框将显示登录失败的提示信息“login failure”。

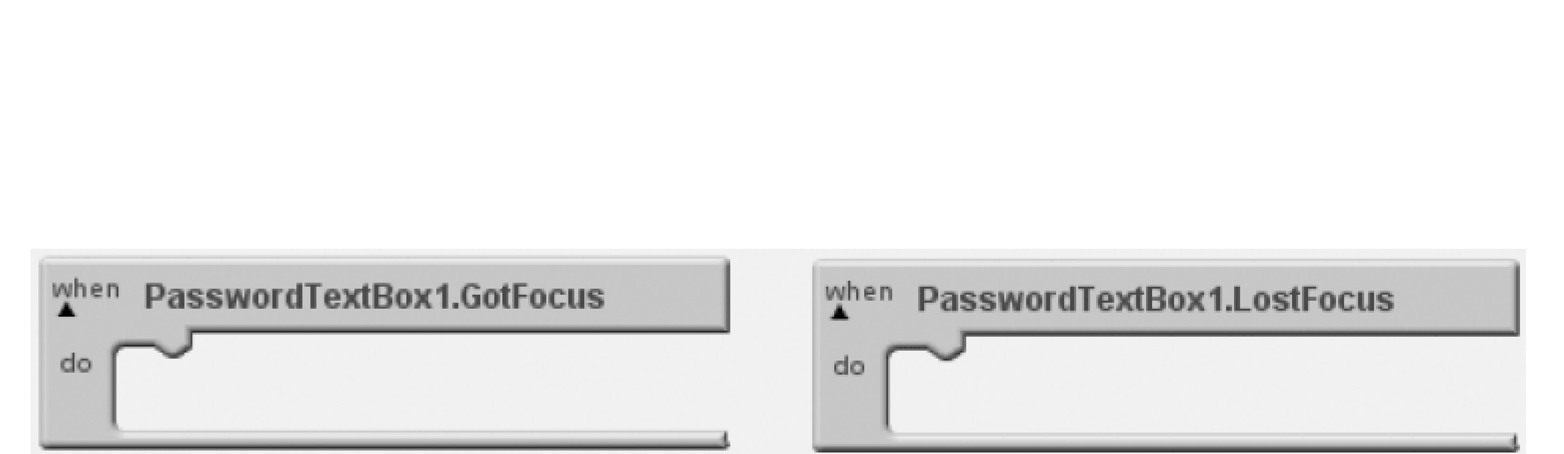


图5.17 密码框控件事件



图5.18 Login示例运行界面

在Login示例的界面设计图(图5.19)上,修改模块区(Components)中的控件名称,尽量使控件名称具有一定的含义,这样便于在模块编辑器中找到需要的事件和属性模块。

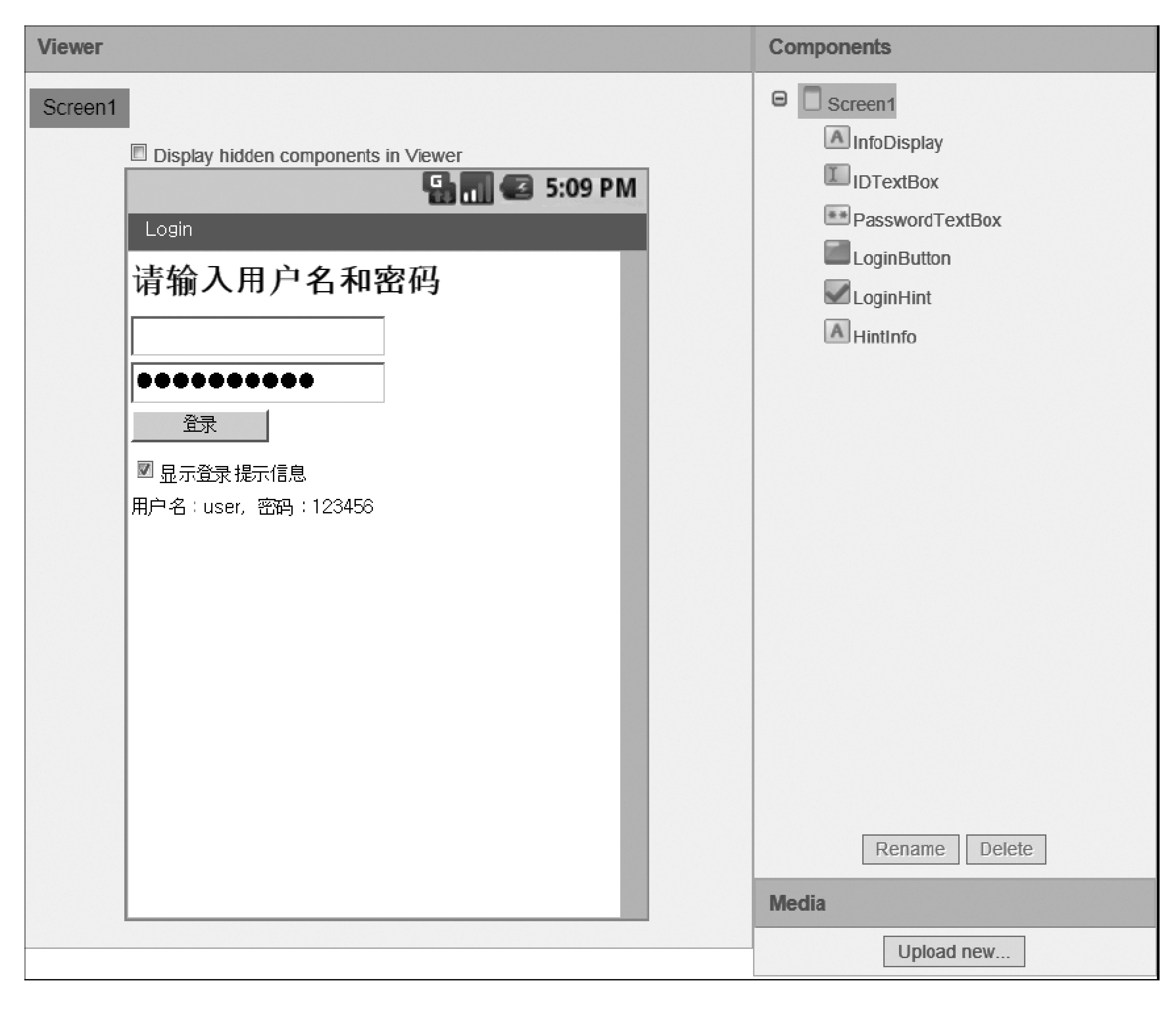


图5.19 Login示例的界面设计图

Login示例的逻辑部分主要响应两个事件，一个是"登录"按钮的单击事件，一个是"显示登录信息"复选框的选项更改事件。

在"登录"按钮的单击事件（图5.20）中，首先判断IDTestBox中的文本是否为"user"，PasswordTextBox中的文本是否为"123456"，这两个判断是用来确定用户名和密码是否与预设值相同。如果用户名和密码正确，在InfoDisplay中显示信息"login success"，否则显示信息"login failure"。

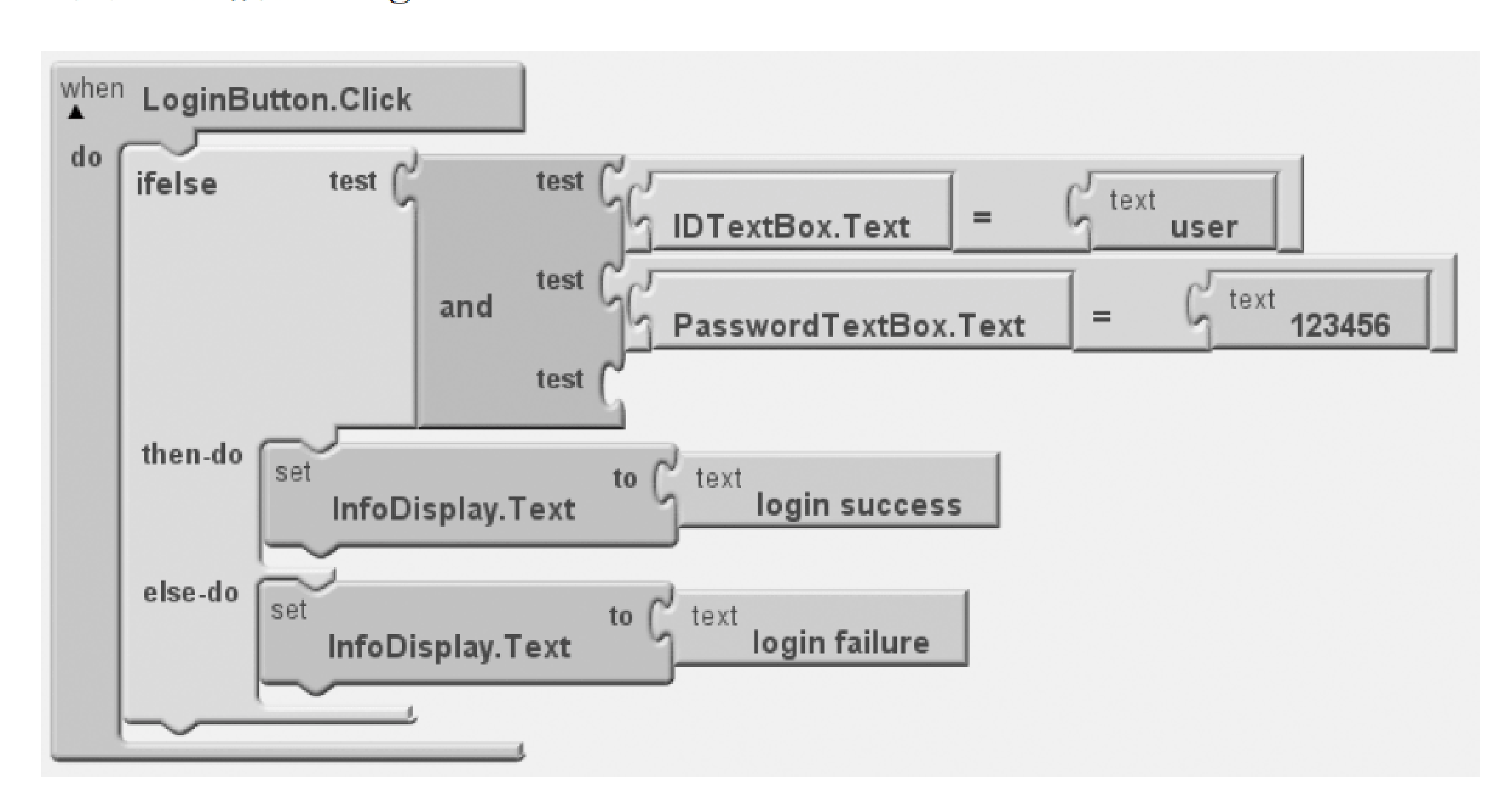


图5.20 "登录"按钮的单击事件

在复选框的选项更改事件（图5.21）中，首先判断复选是否被选中，如果被选中，将HintInfo控件的可见（Visible）属性设为真，显示登录提示信息；反之，将HintInfo控件的可见（Visible）属性设为假，将登录提示信息隐藏起来。

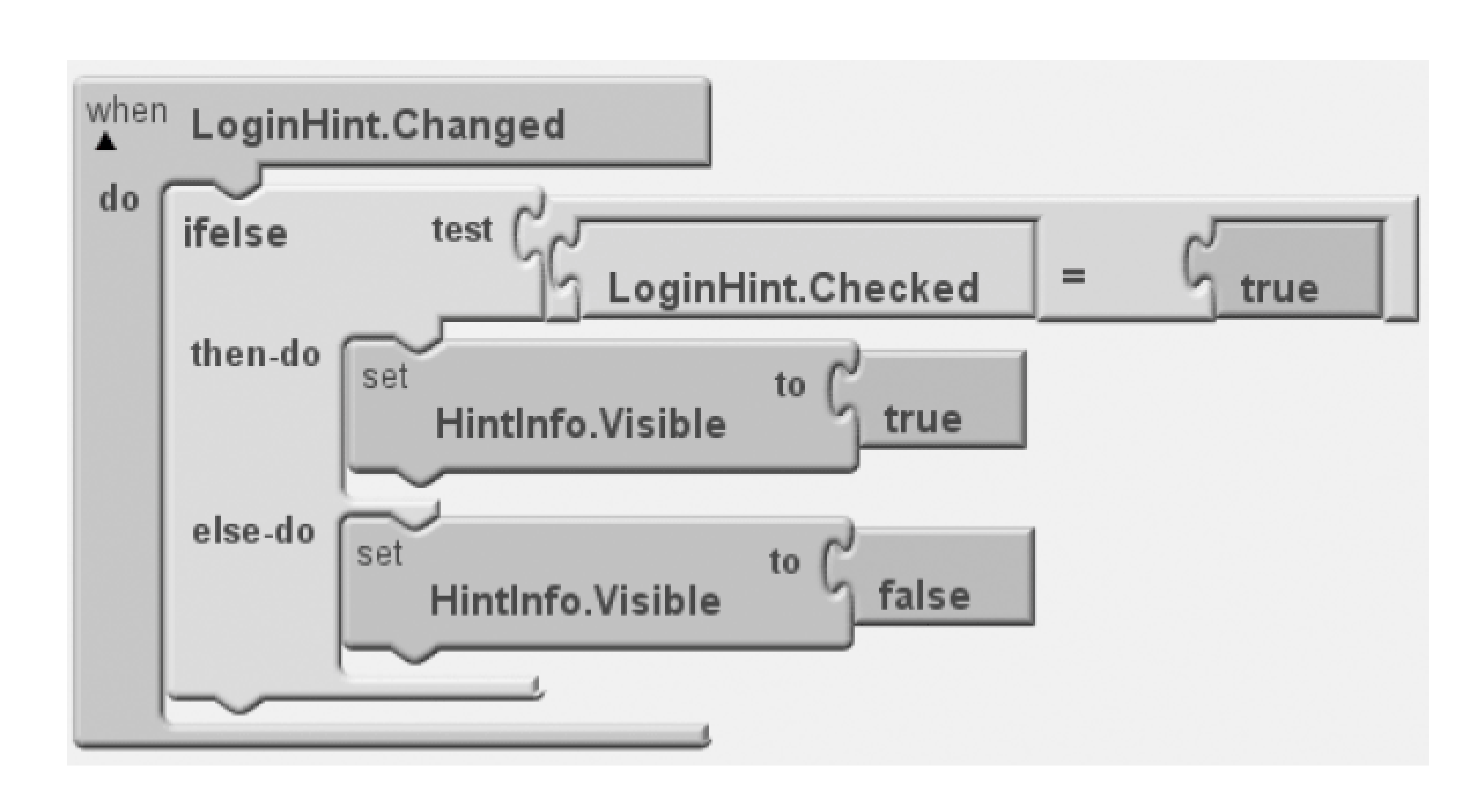


图5.21 "显示登录信息"复选框的选项更改事件

5.2.3 列表选项和时钟

列表选项（ListPicker）是从列表多个选项中选取某一个选项的控件，适合多选一的情况。列表选项被放置在界面上，形状如一般的按钮，如图5.22所示，单击后出现黑色背景的列表供用户选择。在用户单击列表中的某一选项时，黑色背景的列表界面会消失，返回到图5.22的白色列表界面。

列表选项中显示的列表元素，既可以在界面设计器中定义，也可在模块编辑器中进行定义。

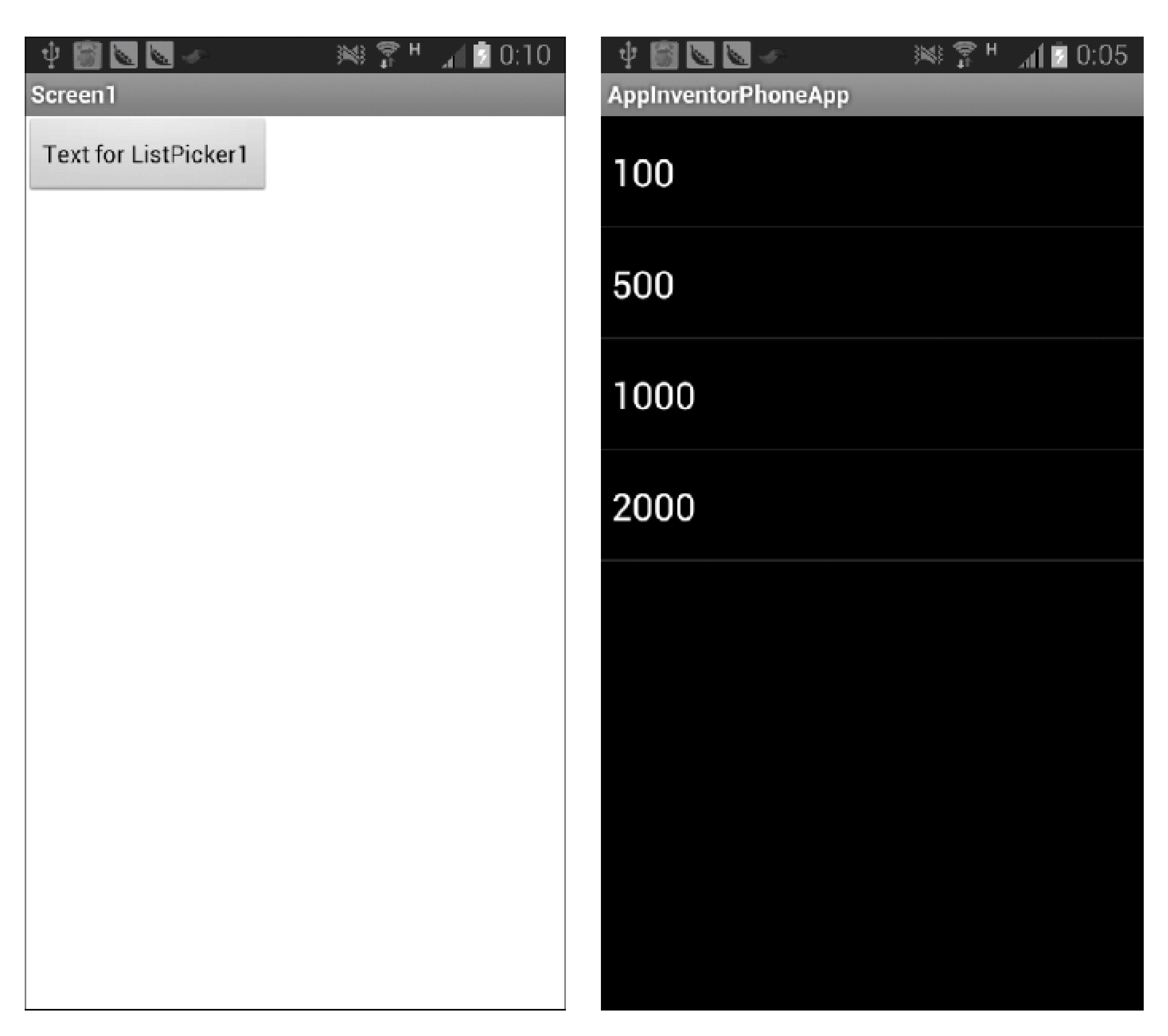


图 5.22　列表选项

在界面设计器中定义列表元素，方法是编辑控件的 ElementsFromString 属性，使用逗号分隔开不同的列表元素。例如，编辑 ElementsFromString 属性为“100，500，1000，2000”，如图 5.23 所示，显示的效果如图 5.22 所示。

另一种方法是在模块编辑器中，将列表选项的 Elements 属性和列表元件（make a list）相关联，并在列表元件内部的 item 槽上添加多个文本（text）数据，完成列表元素的添加，如图 5.24 所示。

ElementsFromString
100, 500, 1000, 2000

图 5.23　列表选项 ElementsFromString 属性

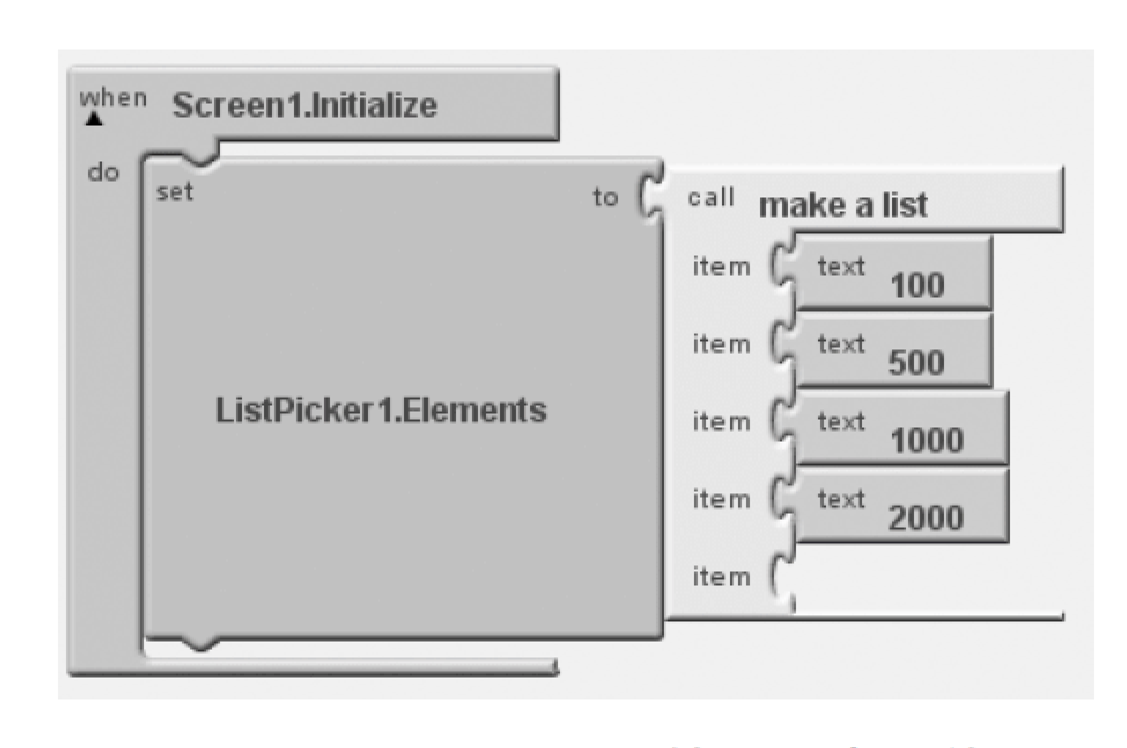


图 5.24　Elements 属性和列表元件

在列表选项的属性中，以往常见的一些属性这里就不再赘述，只介绍列表选项的一些专有属性，如表 5.8 所示。

列表选项的专有事件包括选前操作（BeforePicking）和选后操作（AfterPicking），如图 5.25 所示。

选前操作是在用户单击列表选项的按钮之后发生的事件，这个事件发生在列表元素页面弹出时之前，一般用于处理触发列表选项之后的连带操作。

选后操作是在用户单击选择列表元素之后发生的事件，用于处理用户做出的选择，这是列表选项最常用的事件。

表 5.8　列表选项的专有属性

属　　性	说　　明
Selection	被选中的列表元素
ElementsFromString	字符串方式的列表元素



图 5.25　列表选项专有事件

时钟(Clock)是非可视化组件,可以获取当前时间、格式化输出时间、对时间进行运算,还可以在固定的时间间隔触发事件。由于时钟的应用范围比较广泛,因此被放置在基本控件库当中。

时钟可以很方便地获取当前时间和日期,并可以按照一定的格式显示出来。获取当前时间(手机时间)的方法是调用时钟控件的 Now 方法,可以获取到一个当前的时间点实例(Instant),如图 5.26 所示。

在获取到当前时间点的实例后,就可以调用不同的方法,获取与当前时间点相关的各种显示方式。例如,在图 5.27 中,调用时钟的 FormatDateTime 方法,将 Clock1. Now 拼接在槽 instant 中,就可以输出格式化的"日期+时间",例如 2013-4-1 AM12:00:00。如果调用时钟的 FormatDate 方法,同样将 Clock1. Now 拼接在槽 instant 中,就可以输出格式化的"日期",例如 2013-4-1。

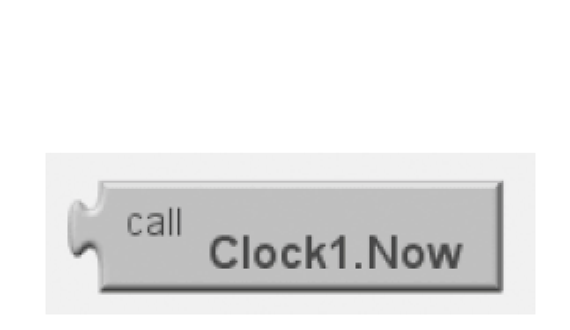


图 5.26　当前时间点

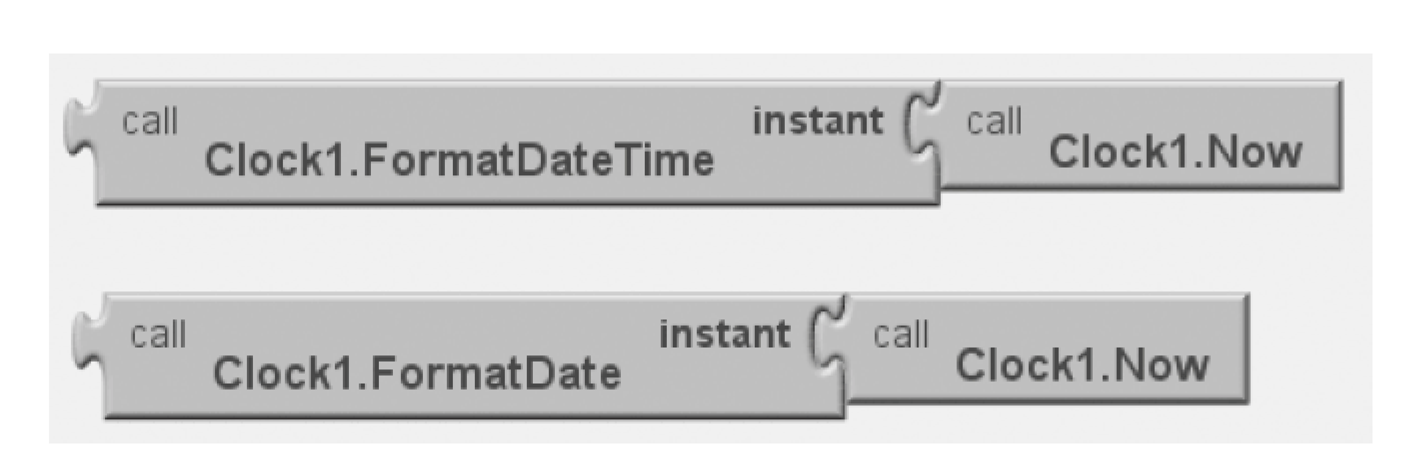


图 5.27　格式化日期和时间

除了可以将当前时间整体显示以外,还可以分别获取当前的年、月、日、小时、分钟、秒、毫秒、星期等信息。获取这些信息都是通过调用时钟的不同方法实现的。在图 5.28 中,分别调用时钟的 Year、Month 和 DayOfMonth 方法,获取当前时间的年、月、日信息。

在实际的开发过程中,"一小时以后"或者"一天以后"这样的时间点也经常被用到。这样的时间点也可也通过时钟的一些方法进行表述。例如在图 5.29 中,通过时钟的



图 5.28　获取当前时间的年、月、日信息

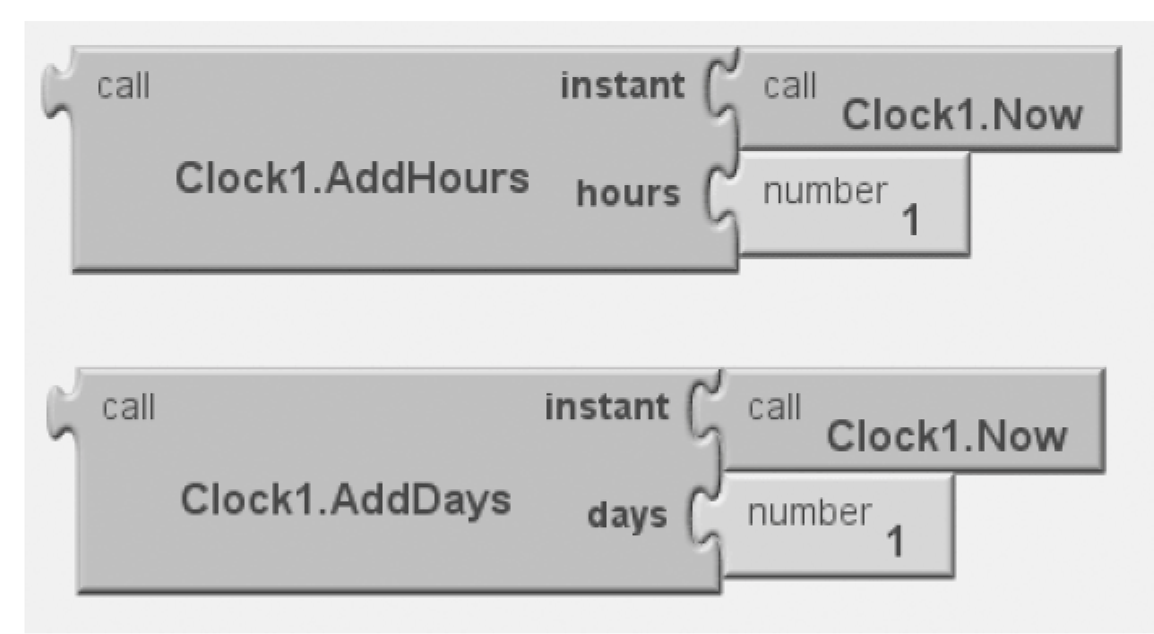


图 5.29　时间计算

AddHours 方法，将 Clock1. Now 拼装在槽 instant 上，并将数字 1 拼装在槽 hours 上，就可以获取一个小时以后的时间点。同样地，将 Clock1. Now 拼装在 AddDays 方法的槽 instant 上，并将数字 1 拼装在槽 days 上，就可以获取一天以后的时间点。这些进行时间计算的方法都是以“Add”开头，例如 AddDays、AddSeconds、AddWeeks 和 AddYears 等。

除了上面所介绍的方法外，时钟还支持一些其他用途的方法，如 Duration、MakeInstantFromMillis 等，时钟的全部方法如表 5.9 所示。

表 5.9　时钟控件所支持的方法

方法名称	说　明
SystemTime	获取手机系统时间，单位微秒
Now	获取当前的时间点
MakeInstant	以“月/日/年 时:分:秒”、“月/日/年”、“时:分”的格式定义时间点
MakeInstantFromMillis(Number millis)	通过毫秒数定义时间点
GetMillis(instant)	从 1970 年到当前时间所经过的毫秒数量
AddSeconds(instant, Number seconds)	计算若干秒以后的时间点
AddMinutes(instant, Number minutes)	计算若干分钟以后的时间点
AddHours(instant, Number hours)	计算若干小时以后的时间点
AddDays(instant, Number days)	计算若干天以后的时间点
AddWeeks(instant, Number weeks)	计算若干星期以后的时间点
AddMonths(instant, Number months)	计算若干月以后的时间点
AddYears(instant, Number years)	计算若干年以后的时间点
Duration(Calendar start, Calendar end)	获取两个时间点的时间差值，单位毫秒
Second(Calendar instant)	获取时间点的秒
Minute(Calendar instant)	获取时间点的分钟
Hour(Calendar instant)	获取时间点的小时
DayOfMonth(Calendar instant)	获取时间点的日期，范围是 1～31 的数字
Weekday(Calendar instant)	获取时间点星期几，范围是 1(周日)～7(周六)的数字
WeekdayName(Calendar instant)	获取时间点星期几，用名称表述
Month(Calendar instant)	获取时间点的月份，用数字表示
MonthName(Calendar instant)	获取时间点的月份，用名称表述
Year(Calendar instant)	获取时间点的年份
FormatDateTime(Calendar instant)	格式化输出日期和时间
FormatDate(Calendar instant)	格式化输出日期
FormatTime(Calendar instant)	格式化输出时间



图 5.30 SuperClock 示例运行界面

下面通过 SuperClock 示例，将前面所介绍过的时间计算、获取显示当前时间点等内容进行展示。SuperClock 示例运行界面如图 5.30 所示，用户单击不同按钮，将在“这里显示时间信息”的标签控件中显示时间信息。

在界面设计图上，可以找到非可视化模块(Non-visible components)中的时钟控件，如图 5.31 所示。

SuperClock 示例的全部逻辑模块如图 5.32 所示。

时钟的另一种用法是作为“定时器”，按照固定的时间间隔反复产生触发事件(Timer)。

时钟有三个相关的属性，如表 5.10 所示。TimerInterval 用来设定触发的时间间隔，单位为毫秒。TimerEnabled 控制着时钟的运行，可以通过修改 TimerEnabled 停止和启动时钟工作。TimerAlwaysFires 是多次产生触发事件开关，如果 TimerAlwaysFires 没有被选中，则产生一次触发事件后，不会继续产生触发事件；反之，则会按照时间间隔不断地产生触发事件。



图 5.31 SuperClock 示例的界面设计图

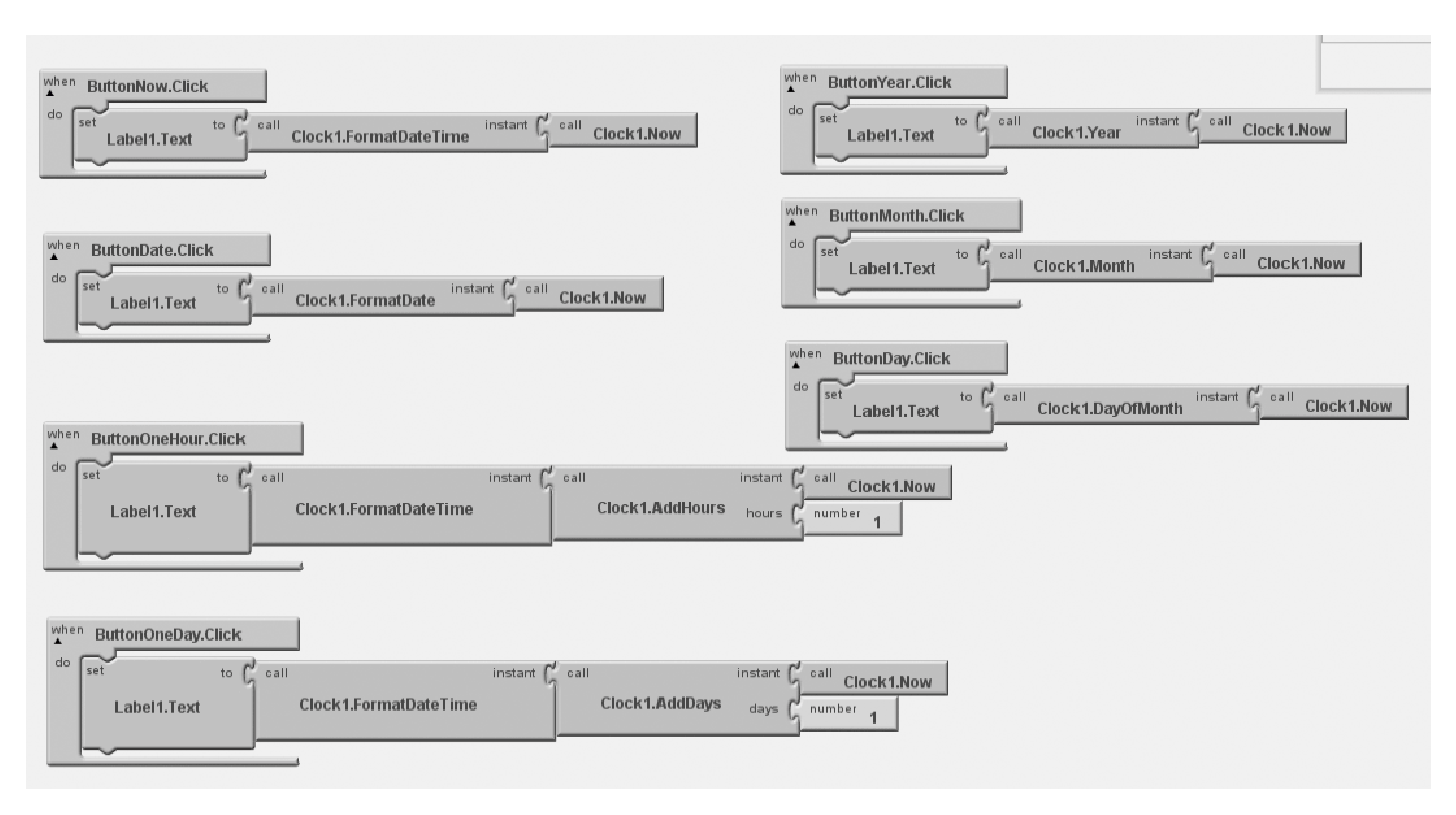


图 5.32 SuperClock 示例全部逻辑模块

时钟只有一个事件：触发事件(Timer)，如图 5.33 所示。在时钟被启动后(TimerEnabled)，经过预定的时间间隔(TimerInterval)后，产生时钟的触发事件。

表 5.10 时钟的属性

模　　式	说　　明
TimerInterval	时间间隔
TimerEnabled	时钟启动开关
TimerAlwaysFires	多次产生定时器事件开关



图 5.33 时钟的触发事件

下面通过 TimerClock 示例介绍如何使用时钟的"定时器"功能，同时也用到了本节介绍过的列表选项。TimerClock 示例的运行界面如图 5.34 所示。(注：TimerClock 示例在程序中使用的沙漏图片素材来源于 http://www.3lian.com/gif/2012/04-24/24667.html。)

在用户单击 Begin 按钮时，时钟开始计时，并在界面上显示计时数字。按钮显示的文字从"Begin"变为"Stop"，如果再次单击按钮，时钟将停止计时。界面下方的"选择时间间隔(单位毫秒)"列表选项，可以控制时钟触发事件的时间间隔。在选择不同的数值后，计数器的累加速度会直接被更改，从而反映在界面中间的数值变化速度上。列表选项可以选择的数字有 100(0.1s)、500(0.5s)、1000(1s)和 2000(2s)。界面设计图如图 5.35 所示。



图 5.34 TimerClock 示例运行界面

TimerClock 示例的逻辑分为 4 个部分，分别是初始化选项列表、选后操作事件、时钟触发事件和按钮单

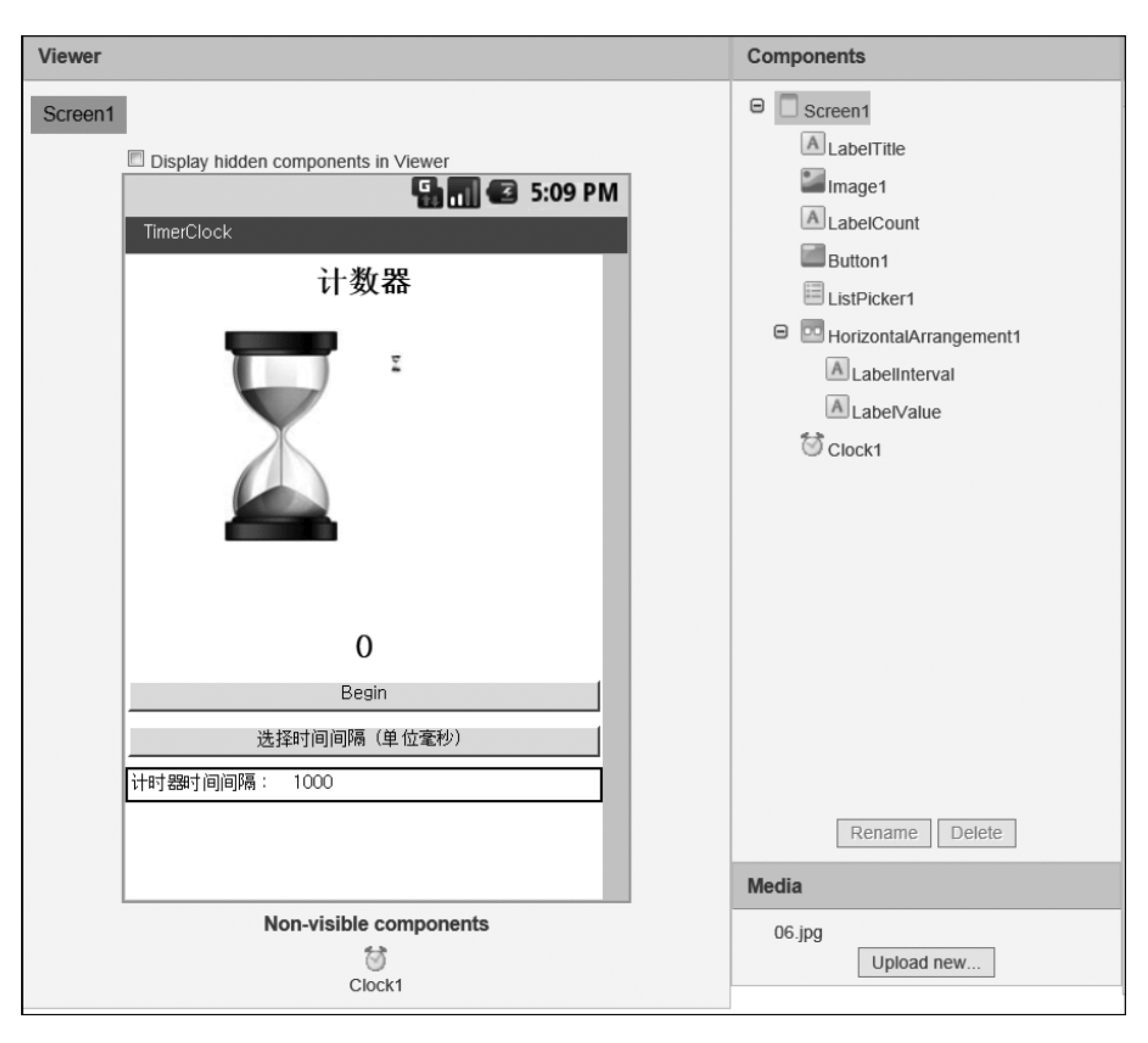


图 5.35 TimerClock 示例的界面设计图

击事件。

第一部逻辑功能是初始化列表选项。在屏幕页 Screen1 的 Initialize 事件中，为列表选项 ListPicker1 的 Elements 属性赋值。赋值的方法是调用 make a list 模块，将文本 100、500、1000 和 2000 拼接在制作列表模块的槽 item 上，如图 5.36 所示。

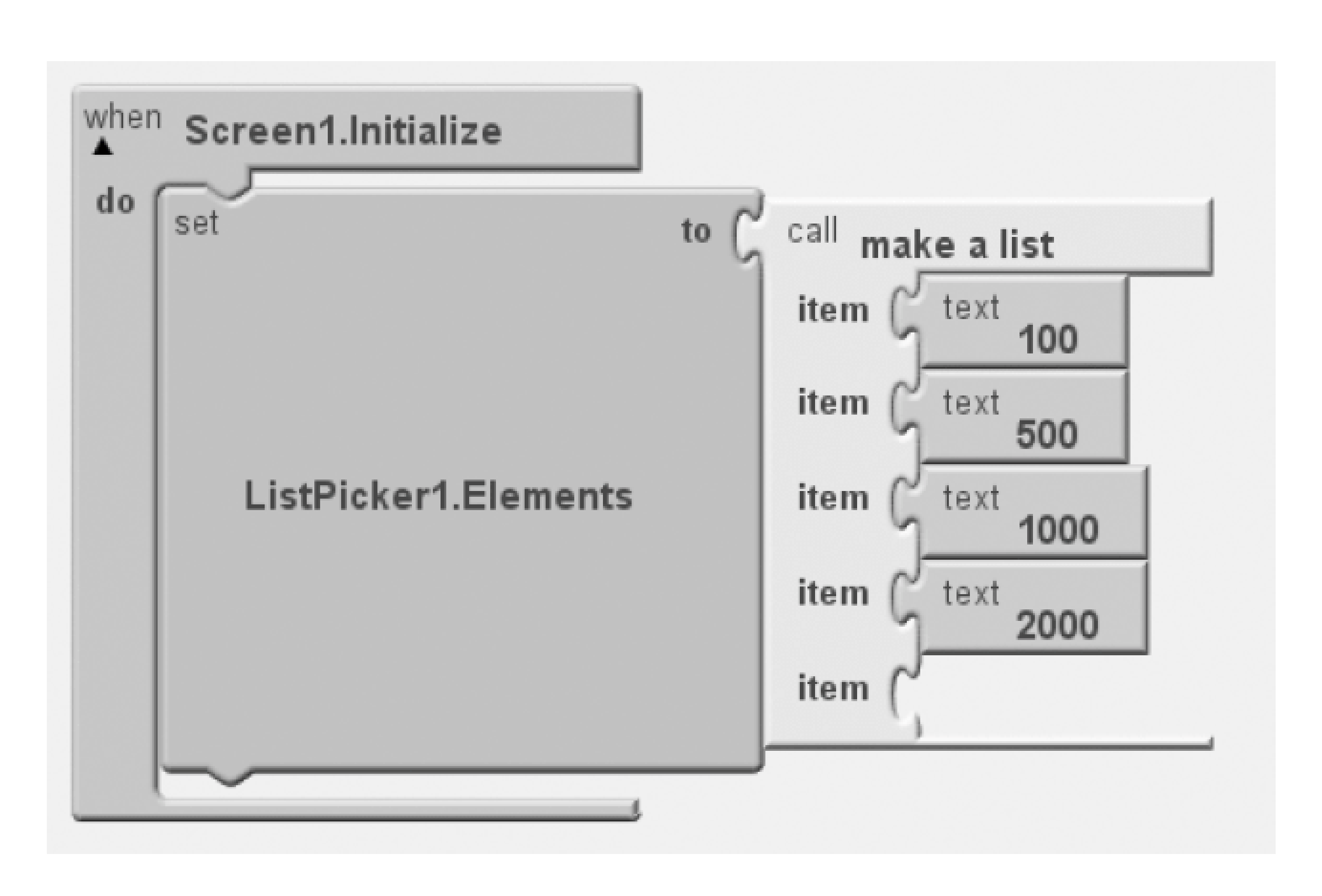


图 5.36 初始化列表

第二部分是列表的选后操作事件的逻辑功能，如图 5.37 所示。在用户选择列表选项的列表元素后，产生选后操作事件，此时需要完成的功能是将用户的选择结果赋值给时钟 Clock1 的 TimerInterval 属性，用来修改时钟的触发时间间隔。同时，将选择结果赋值给标签 LabelValue 的 Text 属性，将选择结果显示在用户界面上。

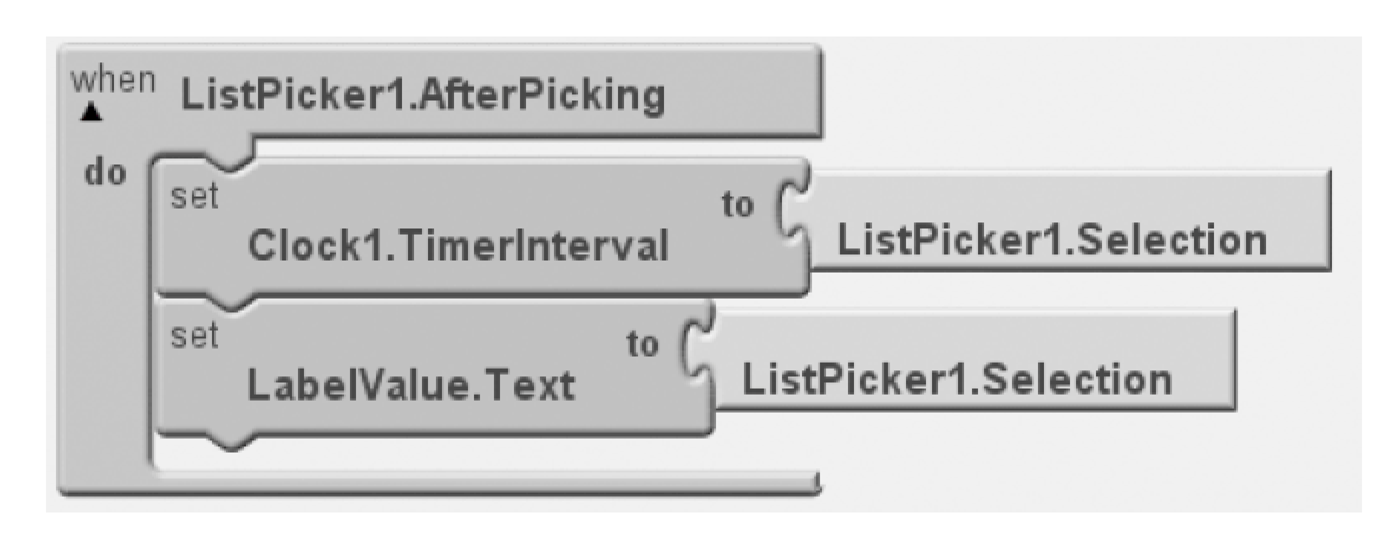


图 5.37 选后操作事件

第三部分是时钟触发事件的逻辑功能，如图 5.38 所示。在每次时钟触发后，修改标签 LabelCount 的 Text 属性，将其在数值上增加 1，在屏幕上会出现“1”、“2”，“3”、“4”、“5”、…的计数信息递增效果。

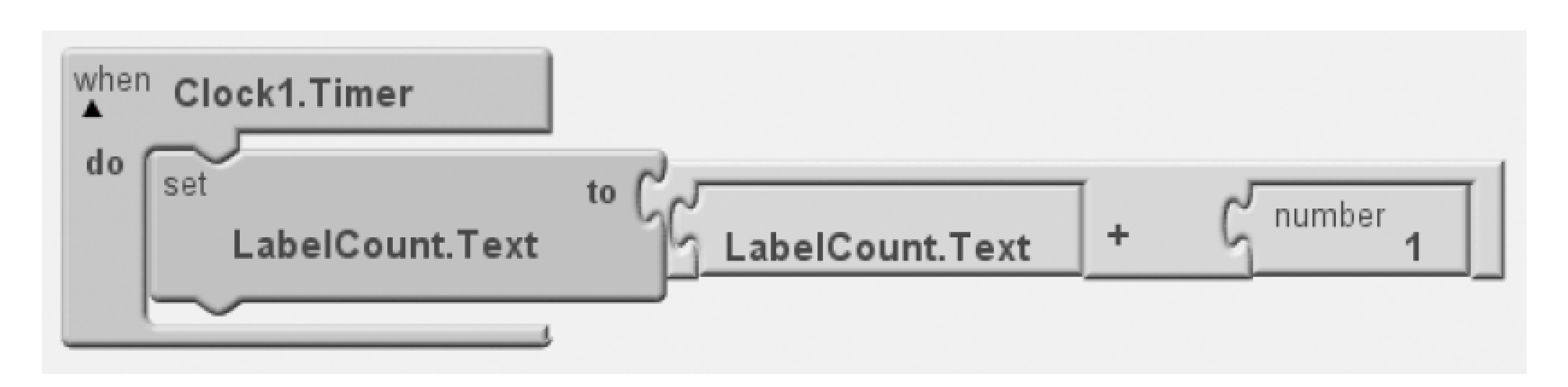


图 5.38 时钟触发事件

第四部分是按钮单击事件的逻辑功能，如图 5.39 所示。

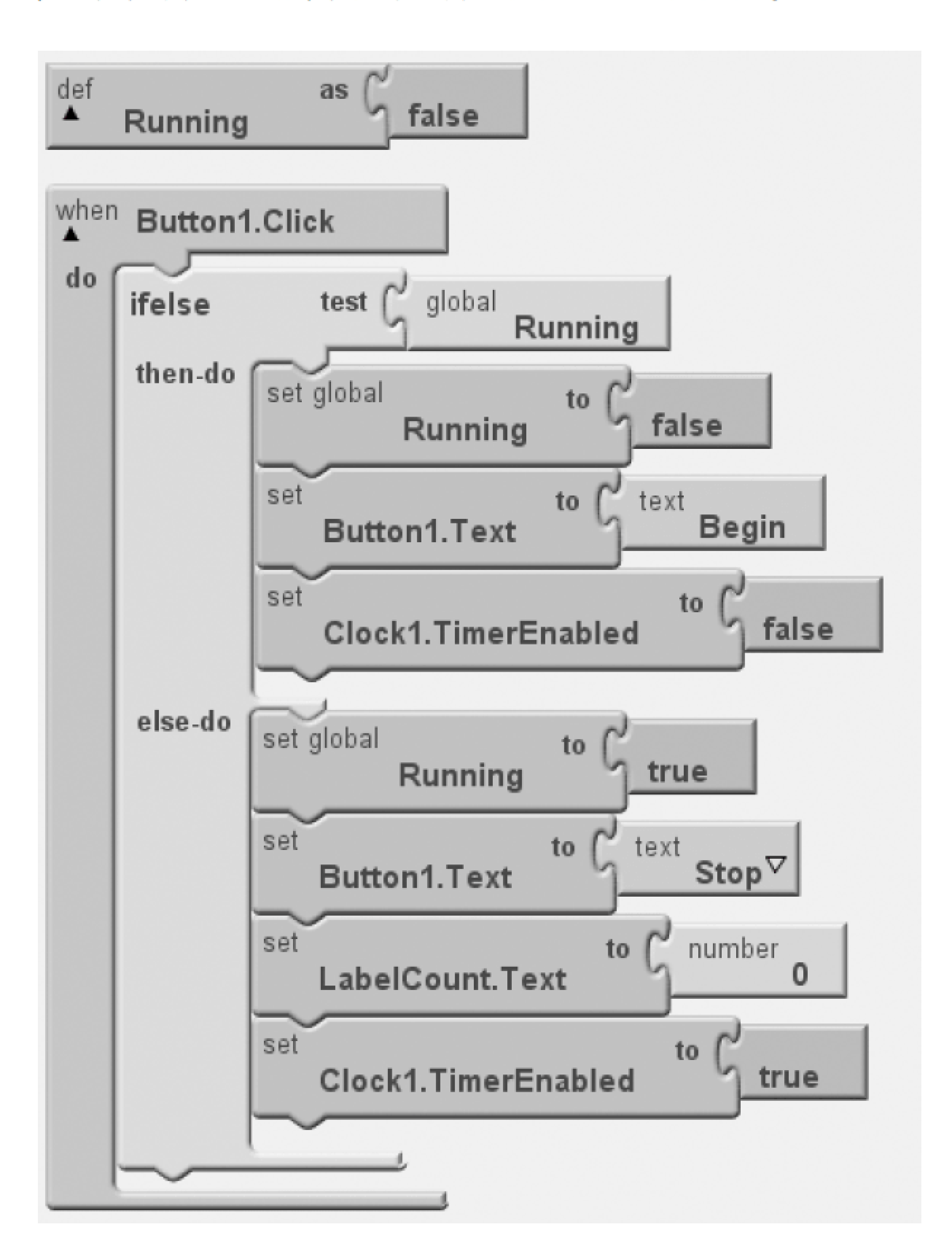


图 5.39 按钮单击事件

首先定义一个全局变量 Running，表示时钟是否开始计数，将其赋值为 false。在按钮 Button1 的单击事件 Click 中，判断全局变量 Running 的值，决定所作的动作。

如果Running为false,表示时钟没有启动,进入else-do分支。先将Running赋值为true;然后将按钮Button1的Text属性赋值为“Stop”,修改按钮显示内容为“Stop”;然后将标签LabelCount的属性Text赋值为0,相当于清空计时器的内容;最后将时钟Clock1的TimerEnabled属性赋值为true,则会立即启动时钟,开始计数过程。

如果Running为true,表示时钟已经启动,进入then-do分支。则先将Running赋值为false;然后将按钮Button1的Text属性赋值为“Begin”,修改按钮显示内容为“Begin”;最后将时钟Clock1的TimerEnabled属性赋值为false,立即停止时钟。

最后给出TimerClock示例的全部逻辑模块,如图5.40所示。

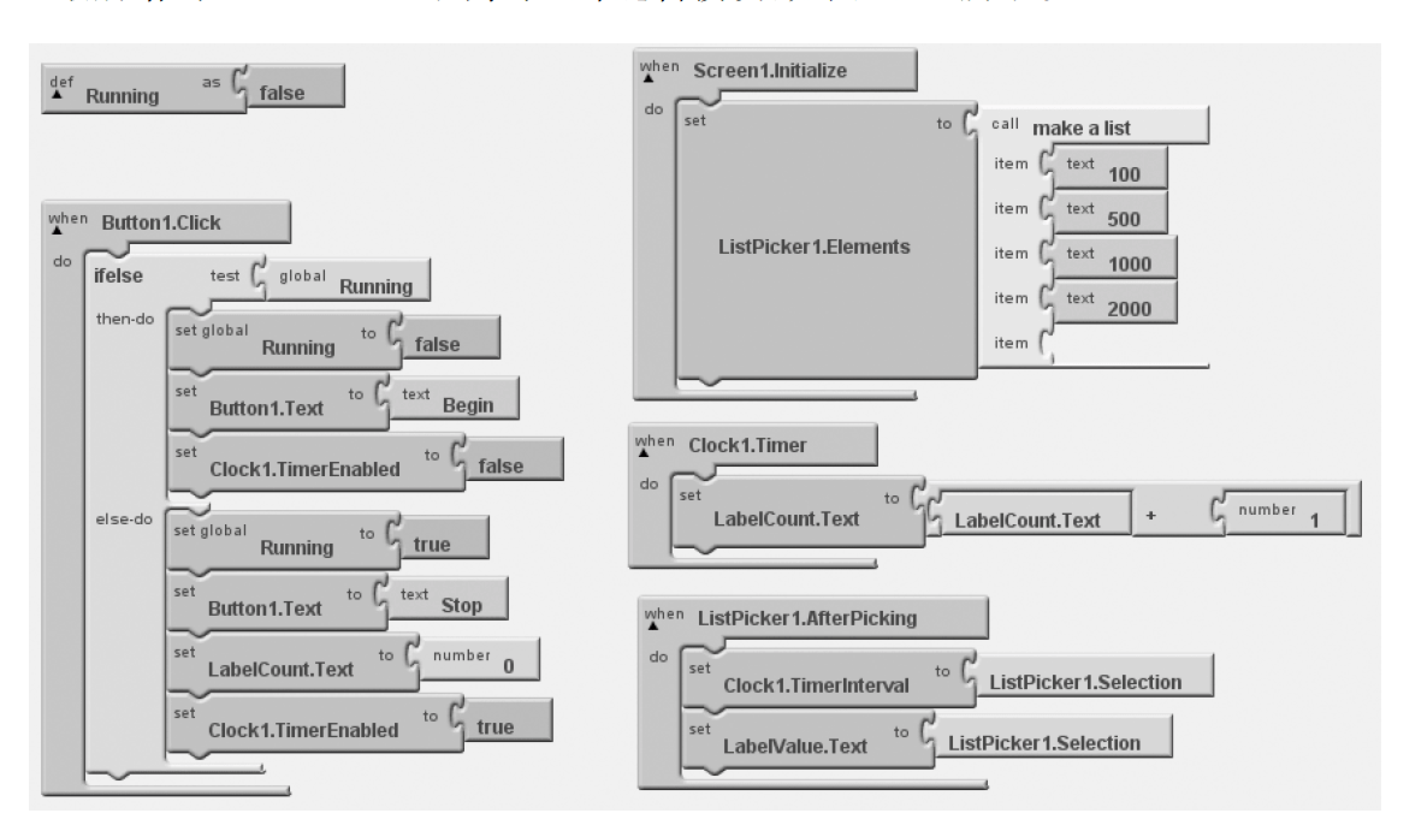


图5.40 TimerClock示例全部逻辑模块

5.3 屏幕布局

在设计应用程序时,往往需要仔细考虑不同控件在界面上的布局和构图效果,而不只是单纯地把控件堆积在屏幕上。为了能够设计出相对美观的布局结构,App Inventor提供了屏幕布局(Screen Arrangement)控件,支持水平布局、垂直布局和表格布局,如图5.41所示。

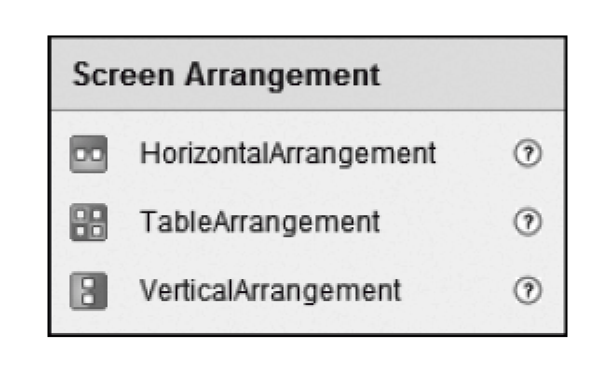


图5.41 可视化控件布局的三种方式

这些布局的作用是按照一定顺序或者相对关系,排列布局内部的其他控件。布局本身在界面上没有任何显示,也不具备事件响应功能,因此布局并不能响应单击或拖动等操作。即便如此,布局仍然是界面设计中不可缺少的组成部分,是提升应用程序视觉效果和美观程度的关键所在。

本节后续的内容将主要介绍布局的使用,并通过使用布局对本章之前的几个示例进行美化和完善。

5.3.1 水平布局

水平布局(HorizontalArrangement)是一种重要的界面布局,也是经常使用到的界面布局。在水平布局中,所有的界面控件都在水平方向上按照顺序进行排列,也就是说,每列仅包含一个控件。

在图 5.42 中,包含两个水平布局,上方的水平布局包含三个复选框控件,“选项 1”、“选项 2”和“选项 3”;下方的布局也包含三个控件,一个标签“提交与否”,两个按钮“提交”和“放弃”。在手机上运行的效果图中,水平布局是不可见的,但在界面编辑器中,可以很容易地找到水平布局的位置,如图 5.43 所示。



图 5.42 水平排布控件的运行效果图



图 5.43 水平排布控件的编辑器效果图

在控件库中将水平布局控件拖曳到界面编辑器后,在界面上形成一个正方形区域,如图 5.44 所示。将第一个控件拖曳到这个正方形的区域中,这个正方形的区域会变成长方形,以后所有加入的控件都会按照水平顺序排布。需要注意的是,不要让水平布局中控件的总尺寸超出屏幕的显示范围。

水平布局只有 Visible、AlignHorizontal、Width 和 Height 4 个属性。Visible 属性表示水平布局中的控件是否可见,如果将其设置为 false,则水平布局中的所有控件都不可见。AlignHorizontal 表示布局中控件的排布方式,支持三种排布方式:左对齐、居中对齐和右对齐。

水平布局的例子可以参考 SuperClock 示例,其中使用了 4 个水平布局,如图 5.45 所示。上方的两个水平布局主要利用了布局的 AlignHorizontal 属性,将其设置为居中对齐(Center),保证图片和显示信息内容居中表示。下方的两个布局,主要用于水平排布按

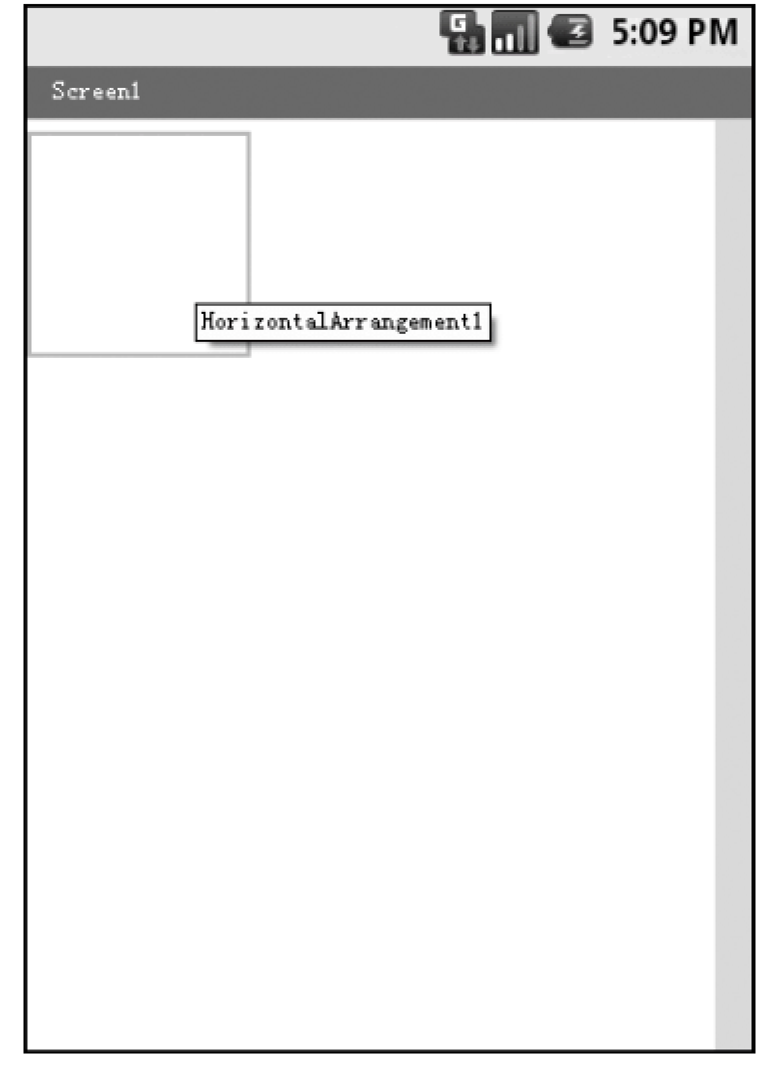


图 5.44 水平布局



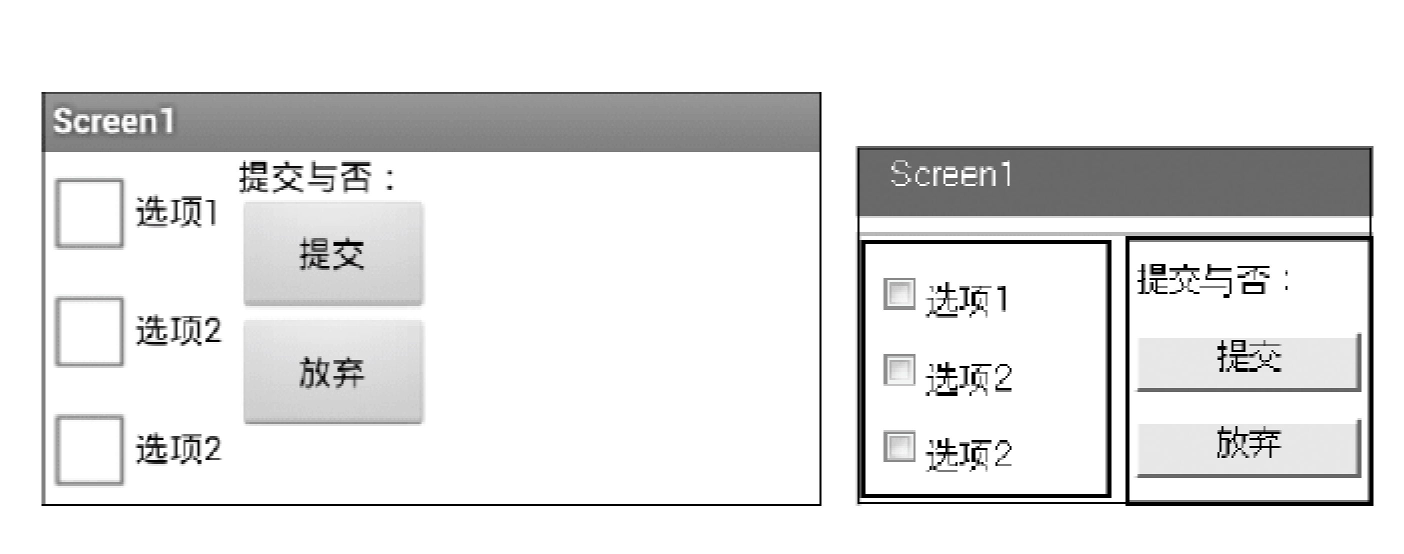
图 5.45 SuperClock 示例

钮控件，AlignHorizontal 属性设置为左对齐，保证所有按钮控件排布在布局的左侧。

5.3.2 垂直布局

垂直布局(VerticalArrangement)中，所有的界面控件都在竖直方向上按照顺序进行排列，也就是说，每行仅包含一个界面控件。

垂直排列控件的运行效果和在界面编辑器中的效果如图 5.46 所示，这里将两个垂直布局嵌套在一个水平布局中，就出现了三个垂直排列的控件组成一组，两个控件组之间是水平排列关系。这样的方式在程序开发中经常会被使用，下面给出 SuperClock 示例界面的垂直布局版本，如图 5.47 所示。



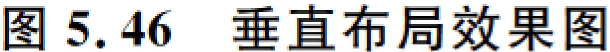

图 5.46 垂直布局效果图



图 5.47 SuperClock 示例(垂直布局版本)

垂直布局的 SuperClock 示例中，7 个按钮被分为三组，组与组之间是水平排布，组内的按钮是垂直排列。例如，第一组中“当前时间”按钮和“一小时以后”按钮就是垂直排列的，第二组中的“当前日期”按钮和“一天以后”按钮也是垂直排列的，但第一组和第二组是水平排列的。

5.3.3 表格布局

表格布局(TableArrangement)也是一种常用的界面布局，它将屏幕划分为表格，通过指定行(Rows)和列(Columns)可以控制格子的数量。图 5.48 是一个 2×2 的表格布局示意图，每个格子可以放置一个控件，当然也可以放置其他布局，如水平布局或垂直布局，实现布局的嵌套。

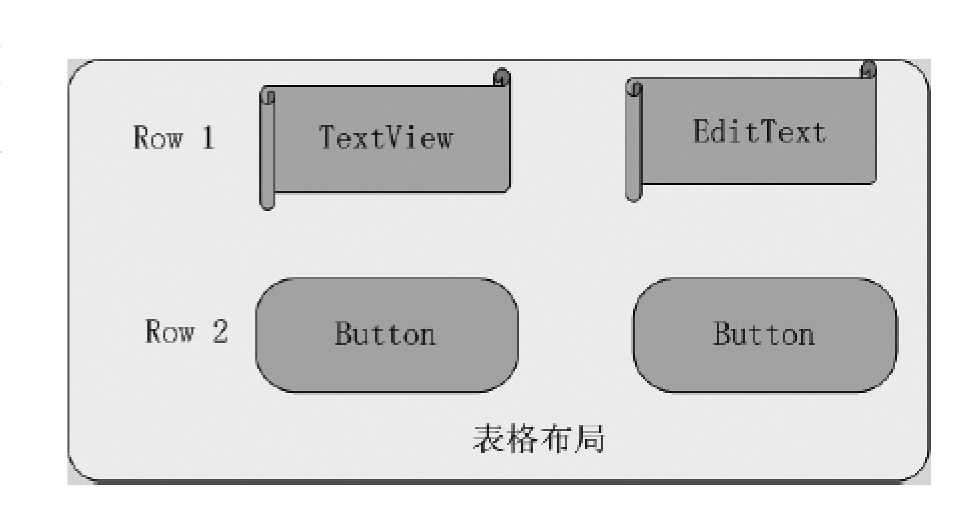


图 5.48 表格布局示意图

行(Rows)和列(Columns)是表格布局的专有属性，表示表格的行数和列数，可以在界面编辑器的属性区内进行更改，如图 5.49 所示。

还是以 SuperClock 示例为蓝本，制作该示例的表格布局版本，如图 5.50 所示。

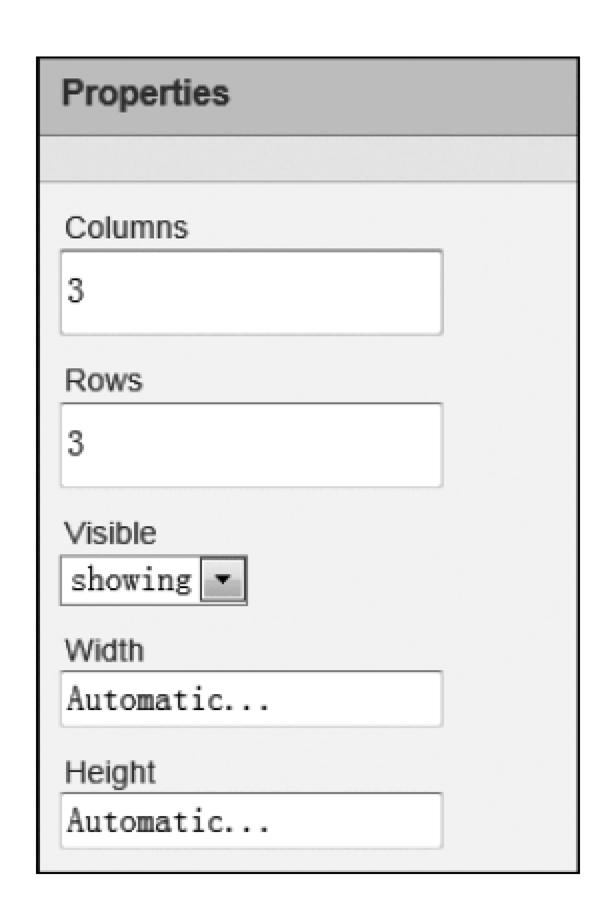


图 5.49 表格布局的属性



图 5.50 SuperClock 示例(表格布局版本)

在表格布局的 SuperClock 示例中,设置了一个 3×3 的表格,将 7 个按钮排布在 9 个格子中。表格布局会根据控件的大小,自动修改表格的大小,这是一个方便、实用的功能。

5.4 媒体控件

随着 Android 应用程序的不断丰富,音频播放、视频的录制与播放、音效播放,以及图片选取等多媒体功能已成为应用开发中不可或缺的部分。App Inventor 的媒体控件支持对音频、视频和图片的操作,全部的媒体控件如图 5.51 所示。

本节所涉及的内容包括录像机、选图工具、音频播放器和视频播放器,相机和音效播放器将在后续章节进行介绍,各媒体控件的名称和功能说明如表 5.11 所示。

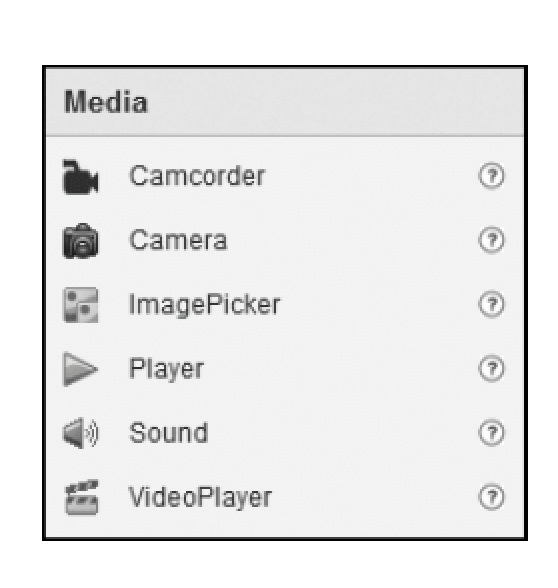


图 5.51 媒体控件

表 5.11 媒体控件功能

控 件	说 明
Camcorder	录像机,用于录制视频
Camera	相机,用于拍摄相片
ImagePicker	选图工具,用于在相册中选取图片
Player	音频播放器,用于播放音频文件或产生手机振动
Sound	音效播放器,用于播放短时间的音效文件
VideoPlayer	视频播放器,用于播放视频

5.4.1 录像机

录像机控件(Camcorder)的功能是利用手机摄像头录制视频。录像控件为非可视化控件,不在界面上显示,也没有可编辑的属性,如图 5.52 所示。

录像机控件简单易用，只提供一个 RecordVideo 方法（录制视频）和一个 AfterRecording 事件（视频录制后），如图 5.53 所示。



图 5.52 非可视化的录像机控件

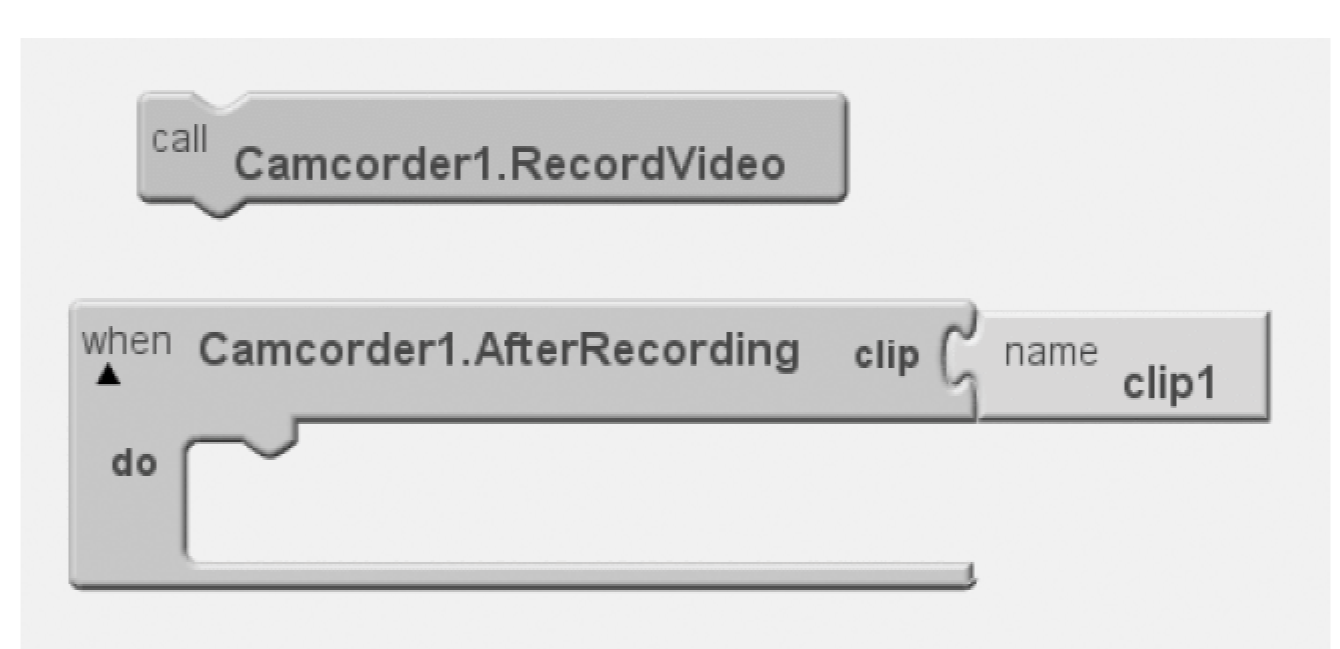


图 5.53 录像机控件支持的方法和事件

RecordVideo 方法启动手机摄像头的录制功能，一般手机的录制过程会被手机内置的“录像（照相）软件”所接管，笔者的手机“录像（照相）软件”如图 5.54 所示。在录制成功后，可以选择“放弃”或“存储”，如果选择“放弃”则这次录制过程将取消，如果选择“存储”，存储路径等信息会在 AfterRecording 事件中被获取到。

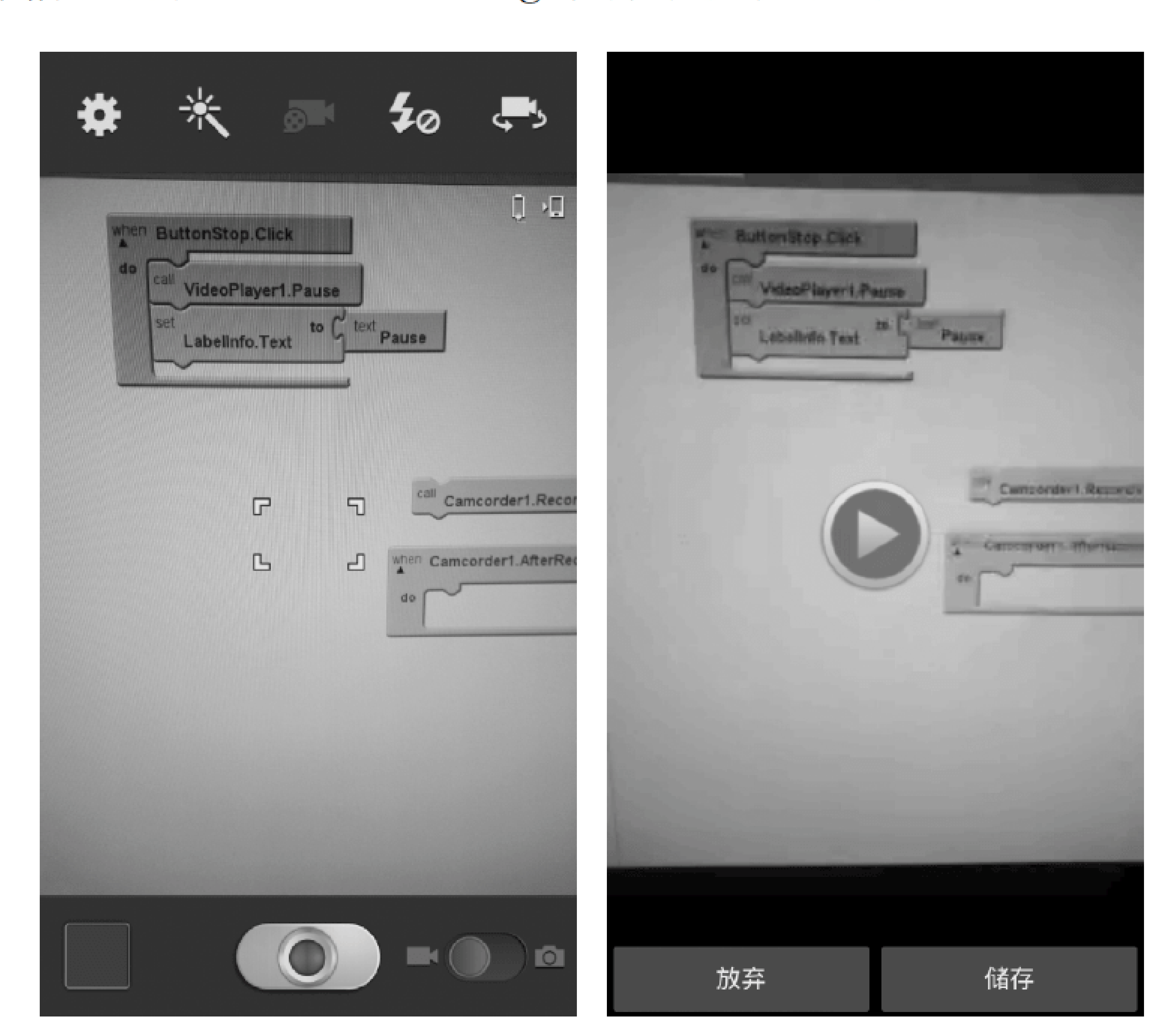


图 5.54 笔者手机内置的“录像（照相）软件”

AfterRecording 事件在录像过程结束时产生，主要功能是提供录像视频的存储路径，信息保存在 clip 参数中。

5.4.2 视频播放器

视频播放器控件（VideoPlayer）主要用于播放视频文件，提供基础的视频播放控制功能，包括播放、暂停、调整播放位置等。视频播放器在界面编辑器中显示为矩形，如图 5.55 所示。

视频播放器中最重要的属性是Source，用来设定所播放的视频文件。视频播放器支持主流的媒体文件类型，包括Windows Media Video(.wmv)、3GPP(.3gp)和MPEG-4(.mp4)。但要注意的是，应用程序中包含视频文件，应通过截取片段或者压缩等方式尽量缩小体积。

视频播放器仅支持Completed(播放完成)事件，在播放到视频文件结尾时产生，Completed事件模块如图5.56所示。

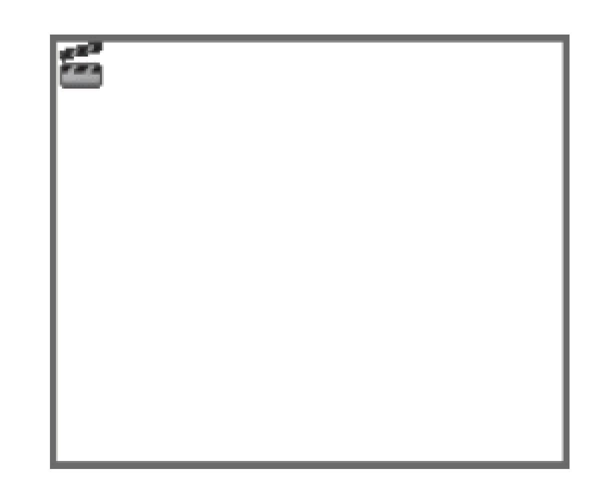


图5.55 界面编辑器中的视频播放器

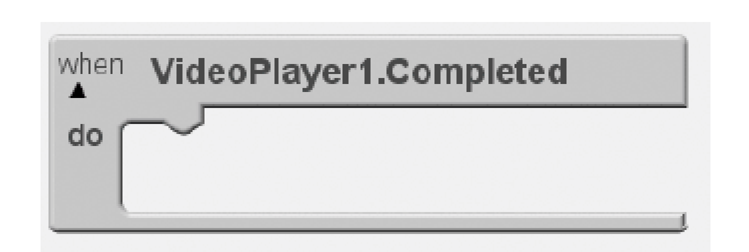


图5.56 Completed(播放完成)事件

视频播放器支持Start、Pause、SeekTo和GetDuration方法，可以调用这些方法控制视频播放过程。视频播放器所支持的方法如表5.12所示。

表5.12 视频播放器的方法

行 为	说 明	行 为	说 明
Start	开始播放视频文件	SeekTo	调整播放位置到所指定时间处(单位毫秒)
Pause	暂停播放当前视频	GetDuration	返回视频的持续时间(单位毫秒)

在视频播放器上单击播放画面，可以弹出“视频控制界面”，使用内置的“视频控制界面”可以在一定程度上减轻开发工作，如图5.57所示。

下面将在RecordPlayer示例中，展示如何使用录像机和视频播放器，制作一个可以使用手机摄像头进行录像，并可以直接播放录像内容的应用程序。RecordPlayer示例的运行界面如图5.58所示。

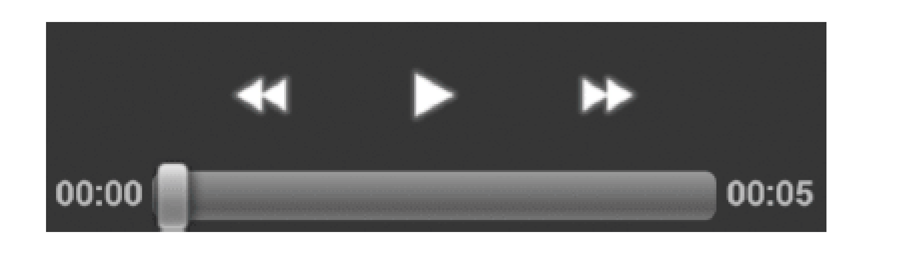


图5.57 视频控制界面



图5.58 RecordPlayer示例运行界面

RecordPlayer 示例界面上的元素简单明了，“摄像”、“播放”和“停止”三个按钮实现了应用程序的主要功能，反馈信息在标签“显示播放器信息”中显示，如图 5.59 所示。

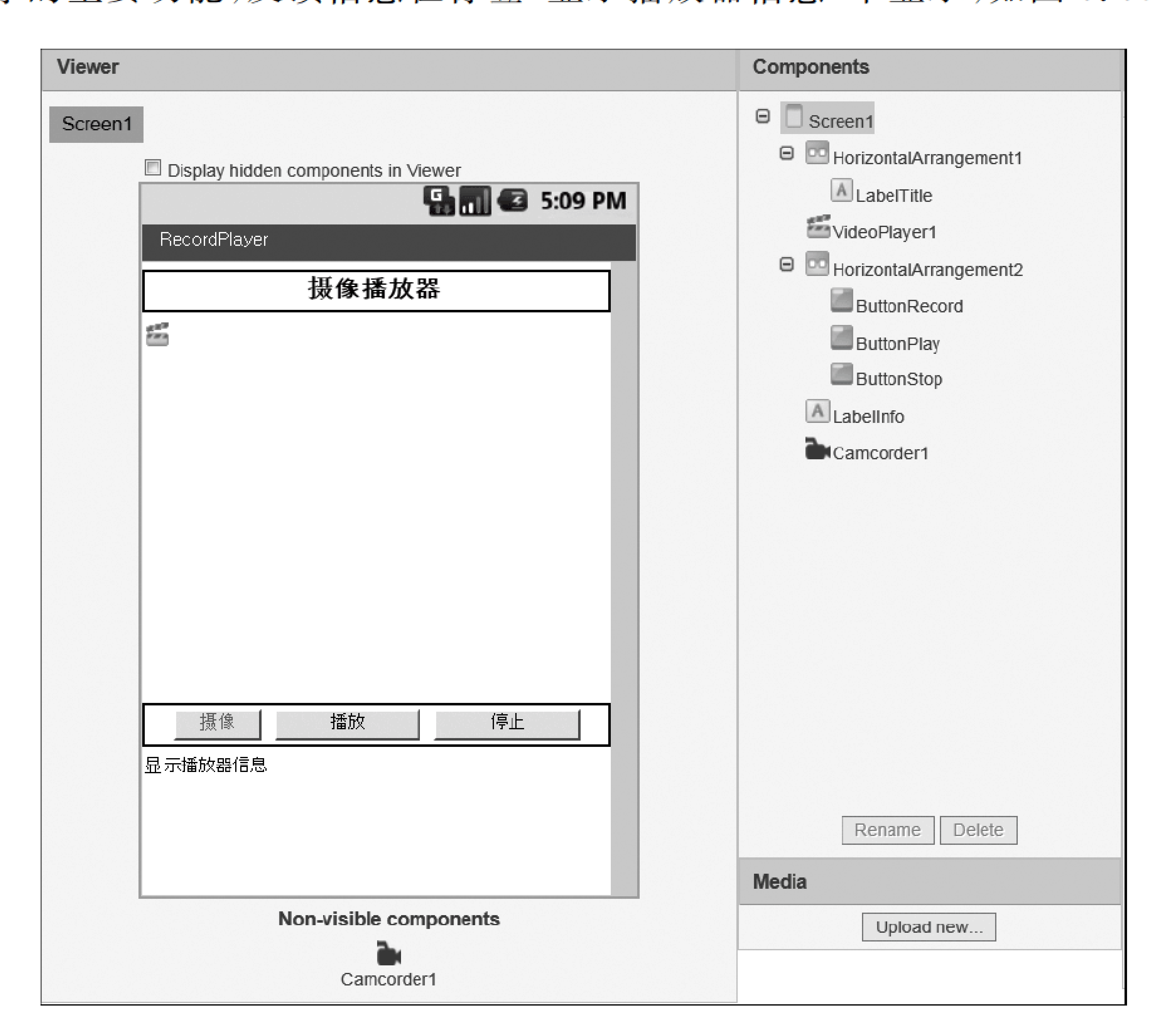


图 5.59 RecordPlayer 示例界面示意图

在 RecordPlayer 示例的界面示意图中，很容易找到非可视化的摄像机控件 Camcorder1 和可见的视频播放器控件 VideoPlayer1，如图 5.60 所示。

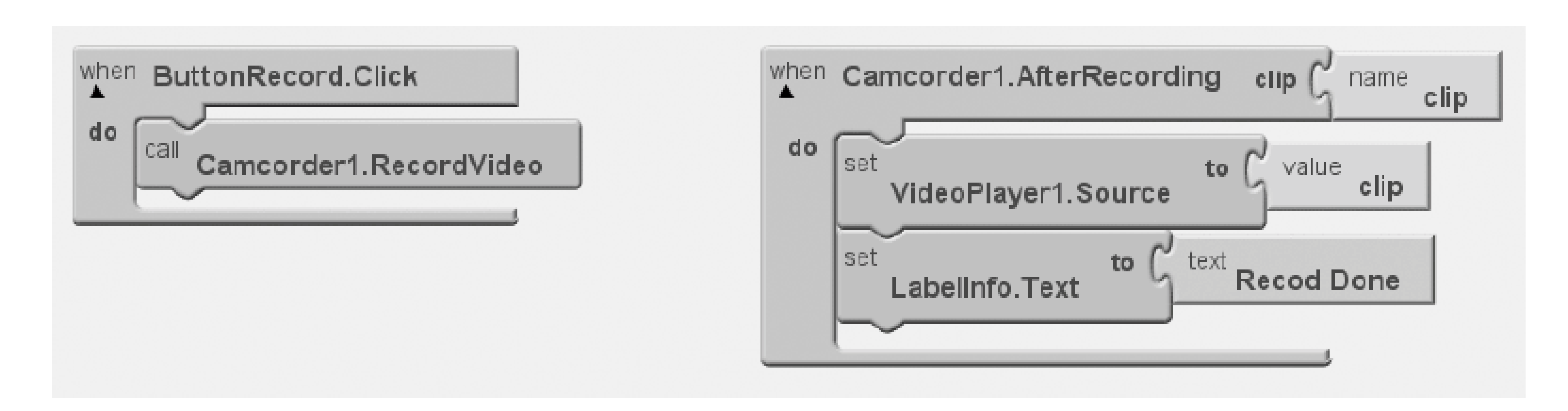


图 5.60 Click 事件和 AfterRecording 事件

逻辑部分首先要在“摄像”(ButtonRecord)按钮的单击事件中调用摄像机 Camcorder1 的 RecordVideo 方法，调用手机的摄像功能。在摄像完毕后(AfterRecording)将视频文件信息传递给视频播放器 VideoPlayer1 的属性 Source。此时，视频播放器 VideoPlayer1 已经获得要播放视频的路径信息，只要调用视频播放器的 Start 方法，就可以播放这个视频文件。

其次，要响应“播放”按钮(ButtonPlay)和“停止”按钮(ButtonStop)的单击事件，如图 5.61 所示。“播放”按钮要调用视频播放器 VideoPlayer1 的 Start 方法播放视频，并将视频的长度信息(GetDuration)显示在标签 LabelInfo 中。“停止”按钮会调用视频播放器 VideoPlayer1 的 Stop 方法暂停视频播放，如果再次单击“播放”按钮，视频将在上次暂停的地方继续播放。

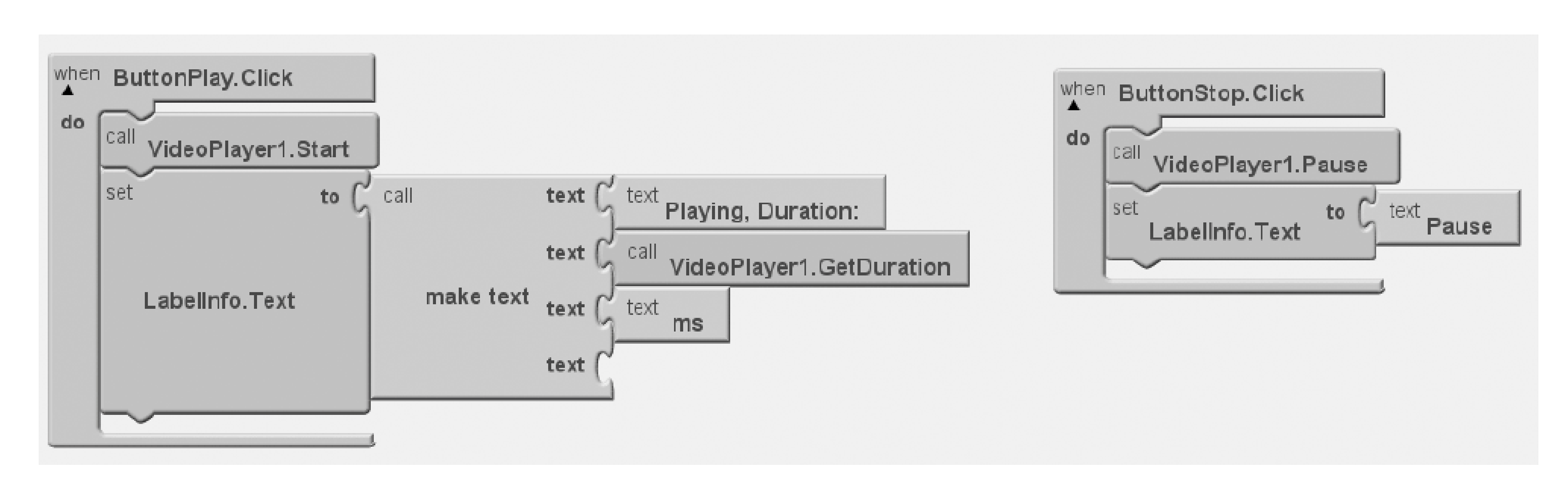


图 5.61　“播放”按钮和“停止”按钮的单击事件

最后就是在视频播放器完成播放后，在标签 LabelInfo 中显示“Play Done”信息。方式是在视频播放器的 Completed 事件中，修改标签 LabelInfo 的 Text 属性，如图 5.62 所示。



图 5.62　视频播放器的 Completed 事件

RecordPlayer 示例的全部逻辑模块如图 5.63 所示。

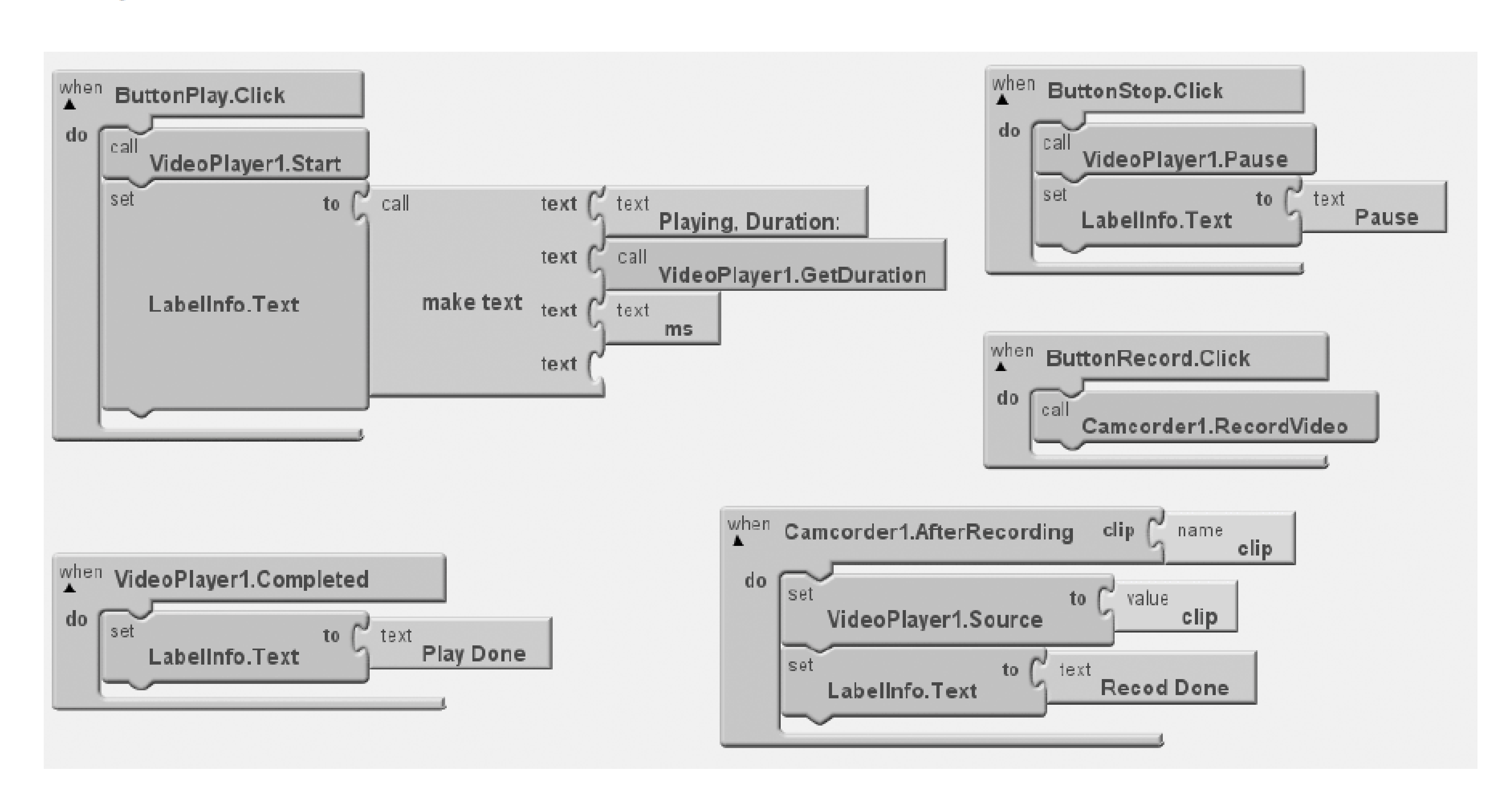


图 5.63　RecordPlayer 示例全部逻辑模块

5.4.3　选图工具

选图工具(ImagePicker)能实现从手机的相册中选取图片的功能。选图工具在界面上的显示外观类似于按钮，如图 5.64 所示，当用户单击选图工具时，就会调用手机内置的“相册软件”，浏览手机内部保存的图片，进而从中选择需要的图片。

Text for ImagePicker1

图 5.64　选图工具

选图工具的 Image 属性用来表示选择的图片，Selection 属性则以字符串的形式保存图片文件的路径信息。

选图工具支持 4 种事件：选图前事件、选图后事件、获取焦点事件和失去焦点事件，如表 5.13 所示。

表 5.13 选图工具事件

事　　件	说　　明	事　　件	说　　明
BeforePicking	选图前事件	GotFocus	获取焦点事件
AfterPicking	选图后事件	LostFocus	失去焦点事件

选图工具只支持 Open 方法，用来打开手机内置的“相册软件”。

5.4.4 音频播放器

音频播放器(Player)用来播放音频文件，一般用于播放时间较长的音频文件，如音乐、录音等。如果时间较短的音频，如铃声、提示音，则多用音效播放器进行播放，因为音效播放器的资源消耗较少。

音频播放器除了具有音频播放功能外，还可以让手机产生振动。音频播放器属于非可视化控件，并不在界面内显示。在界面编辑器中的音频播放器如图 5.65 所示。

音频播放器支持 4 个属性：IsLooping、IsPlaying、Source 和 Volume。Source 属性用来定义音频文件源，可以是上传的音频文件，也可以是网络中的音频文件。音频播放器支持常见的 mp3、wav、mid 和 3gp 等音频格式。

Volume 表示播放音量，是 0～100 的整数，数值越大，音量越大。音频播放的属性如表 5.14 所示。

音频播放器唯一支持的事件是 Completed，如图 5.66 所示。Completed 事件在音频播放完成后产生，通常用于执行播放完毕后的下一步动作。



图 5.65 音频播放器

表 5.14 音频播放的属性

属　　性	说　　明
IsLooping	是否循环播放
IsPlaying	是否正在播放
Source	音频文件源
Volume	播放音量



图 5.66 Completed 事件

音频播放器支持 Start、Pause、Stop 和 Vibrate 4 种方法，用于开始和暂停音频播放，以及控制手机振动。音频播放器支持的全部方法如表 5.15 所示。

表 5.15 音频播放器的方法

方　　法	描　　述	方　　法	描　　述
Pause	暂停当前播放	Stop	停止播放
Start	开始播放	Vibrate	手机振动

Vibrate 方法会让手机产生一段时间的振动，振动时间由参数 milliseconds 控制，单位是毫秒，如图 5.67 所示。

下面将通过 MusicGallery 示例，尝试如何使用选图工具和音乐播放器控件设计应用

程序。MusicGallery 示例实现的功能类似于音乐相册，在播放背景音乐的同时，循环显示用户选择的两张图片。

MusicGallery 示例的运行界面如图 5.68 所示。用户需要单击“选取照片”按钮两次，利用选图工具在手机内置的图片库中选取两张图像文件。在选前第一张图片前，界面上显示红色的提示语“Please pick first picture!”，在用户选择好第一张图片后，提示语变为“Please pick second picture!”。在用户将两个图片都选好后，提示语将变为蓝色“Ready!”，表示可以单击“启动相册”按钮，开始真正的“音乐相册”之旅。在用户未单击“停止相册”按钮前，将持续播放音乐和循环显示图片。

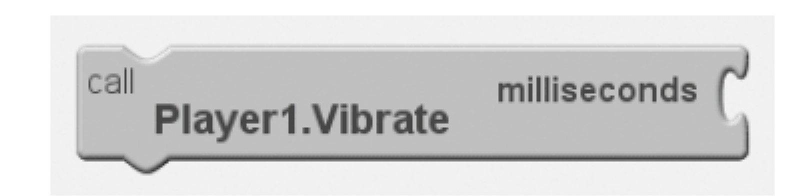


图 5.67 Vibrate 方法



图 5.68 MusicGallery 示例的运行界面

MusicGallery 示例的界面设计图如图 5.69 所示。



图 5.69 MusicGallery 示例的界面设计图

在界面设计过程中，要注意将“启动相册”和“停止相册”按钮的 Enabled 项设为 false，只允许用户在选择完两张图片后才可以启动相册。

将时钟 Clock1 的 Enabled 项设为 false，让时钟在用户单击“启动相册”按钮以后才启动，因为时钟的主要作用是周期性地切换图片。Clock1 的时间间隔(TimerInterval)设置为 10 000ms(10s)，TimeAlwaysFires 设置为 true。

音频播放器 Player1 的 Source 属性设置为已经上传的 MIDI 音频文件，读者也可以选择自己喜欢的音频，但需要控制文件的大小，避免资源文件的上传时间过长。音频播放器的 IsLooping 属性应设置为 true，因为这里需要无限循环播放背景音乐。但笔者发现在界面编辑器中设置的 IsLooping 属性并不生效，需要在逻辑编辑器中，将 IsLooping 属性初始化工作放在屏幕页 Screen1 的 Initialize 事件中完成才能生效，如图 5.70 所示。

在逻辑编辑器中，首先定义三个全局变量 Status、Picture1 和 Picture2，如图 5.71 所示。Picture1 和 Picture2 表示用户选择的两个图像。

Status 则表示程序所处的状态。当程序刚刚初始化完毕，用户还没有进行任何操作时，程序处于的状态为 0；当用户选择完第一张图片后，程序处于的状态为 1；当用户选择完第二张图片后，程序处于的状态为 2。全局变量 Status 以此来标识用户是否已将两张图片选取好了。用户单击“启动相册”按钮后，时钟触发事件会将 Status 在 1 和 2 之间切换，此时的全局变量 Status 主要用来控制切换显示图像。在用户单击“停止相册”按钮后，Status 变为 0。

用户操作的第一步是要选择两张图片，在选图工具 ImagePicker1 的 AfterPicking 事件中，将选择的图片分别赋值给全局变量 Picture1 和 Picture2，并修改提示信息的内容和颜色。在完成两张图片的选取后，通过修改“启动相册”按钮的 Enable 属性为 true，允许用户单击该按钮。ImagePicker1 的 AfterPicking 事件的逻辑功能如图 5.72 所示。



图 5.70　屏幕页 Screen1 的 Initialize 事件

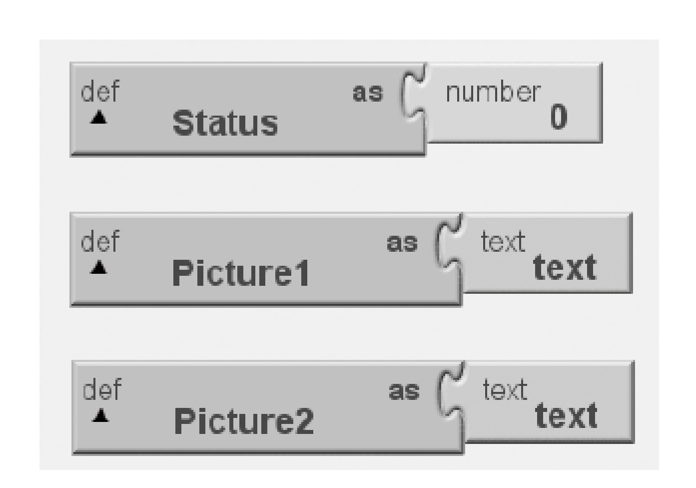


图 5.71　定义全局变量

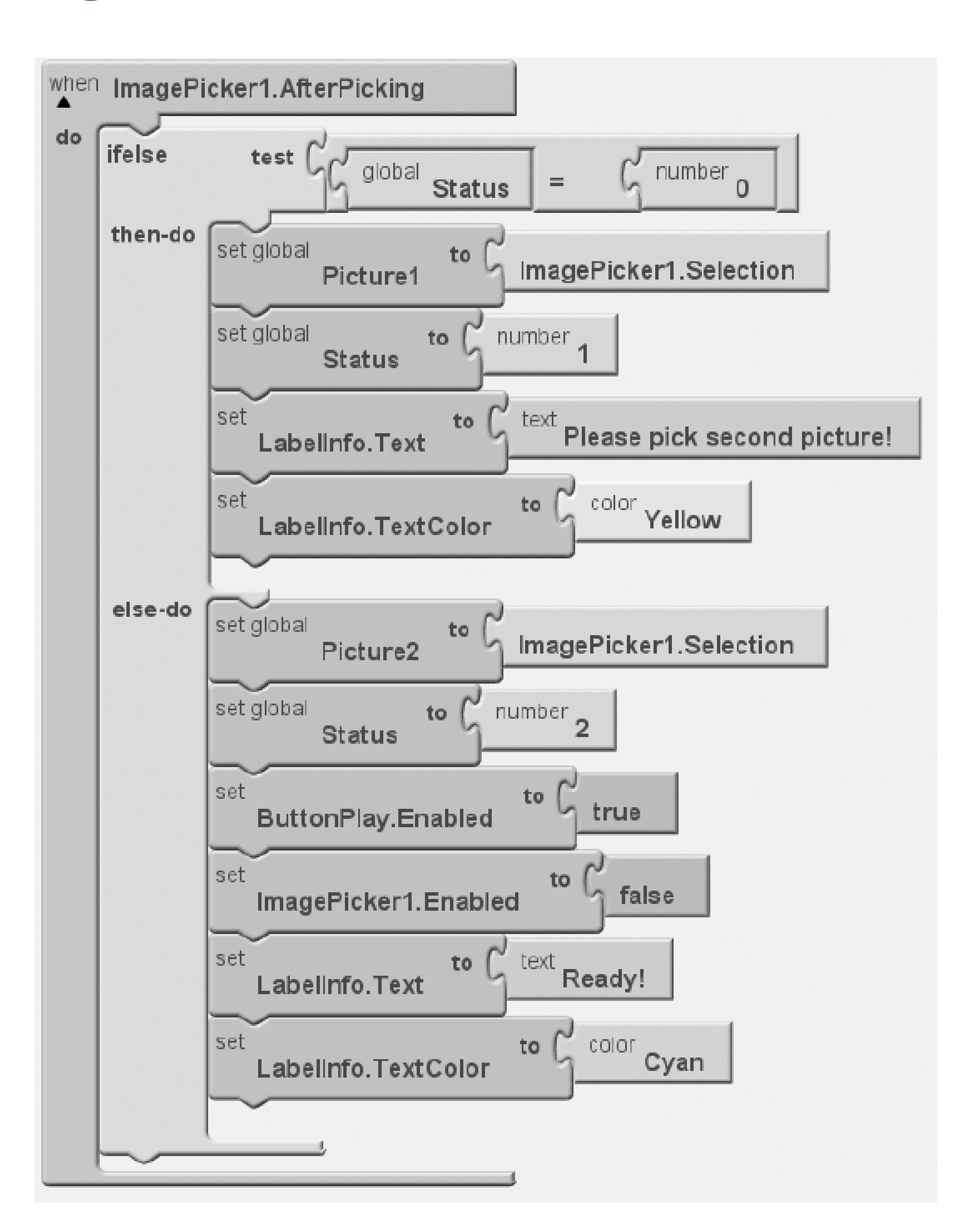


图 5.72　ImagePicker1 的 AfterPicking 事件

用户操作的第二步是通过单击“启动相册”按钮，启动音频播放器 Player1 音频播放功能，这里通过调用 Player1 的 Start 方法实现。将全局变量 Picture1 赋给图像控件 Image1 的 Picture 属性，这样就可以在图像控件中显示用户选择的第一幅图片。

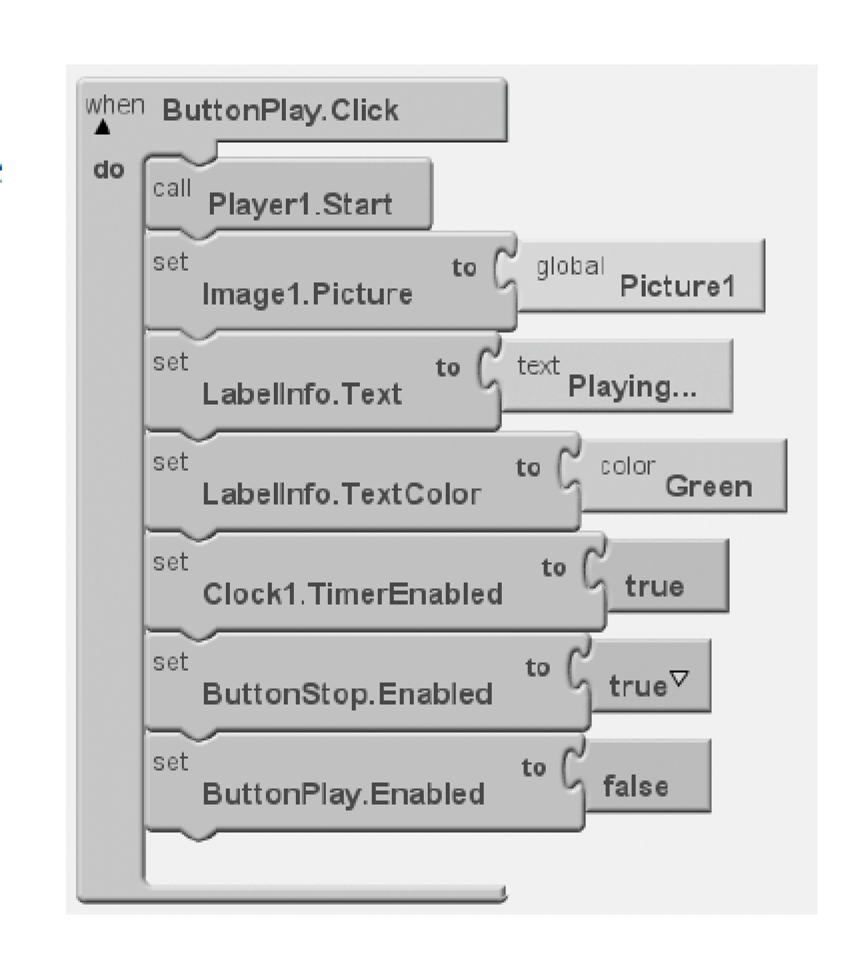


图 5.73 ButtonPlay 按钮的 Click 事件

在“启动相册”按钮的单击事件中，还需要修改提示信息的内容和颜色，并将自身设置为灰色的不可单击状态（Enabled 属性为 false），将“停止相册”按钮设置为可以单击的状态（Enabled 属性为 true）。

还有一件非常重要的工作，就是启动时钟，方法依旧是将时钟 Clock1 的 TimerEnabled 属性设置为 true。

ButtonPlay 按钮的 Click 事件的逻辑如图 5.73 所示。

因为 TimeAlwaysFires 属性被设置为 true，所以时钟 Clock1 启动后会持续产生 Timer 事件。在 Timer 事件中，通过修改全局变量 Status 的值，切换显示第一张图或第二张图，切换的时间间隔是 10s，如图 5.74 所示。

用户操作的第三步是通过单击“停止相册”按钮，停止音乐播放，停止图片切换和暂停时钟 Clock1，逻辑功能如图 5.75 所示。

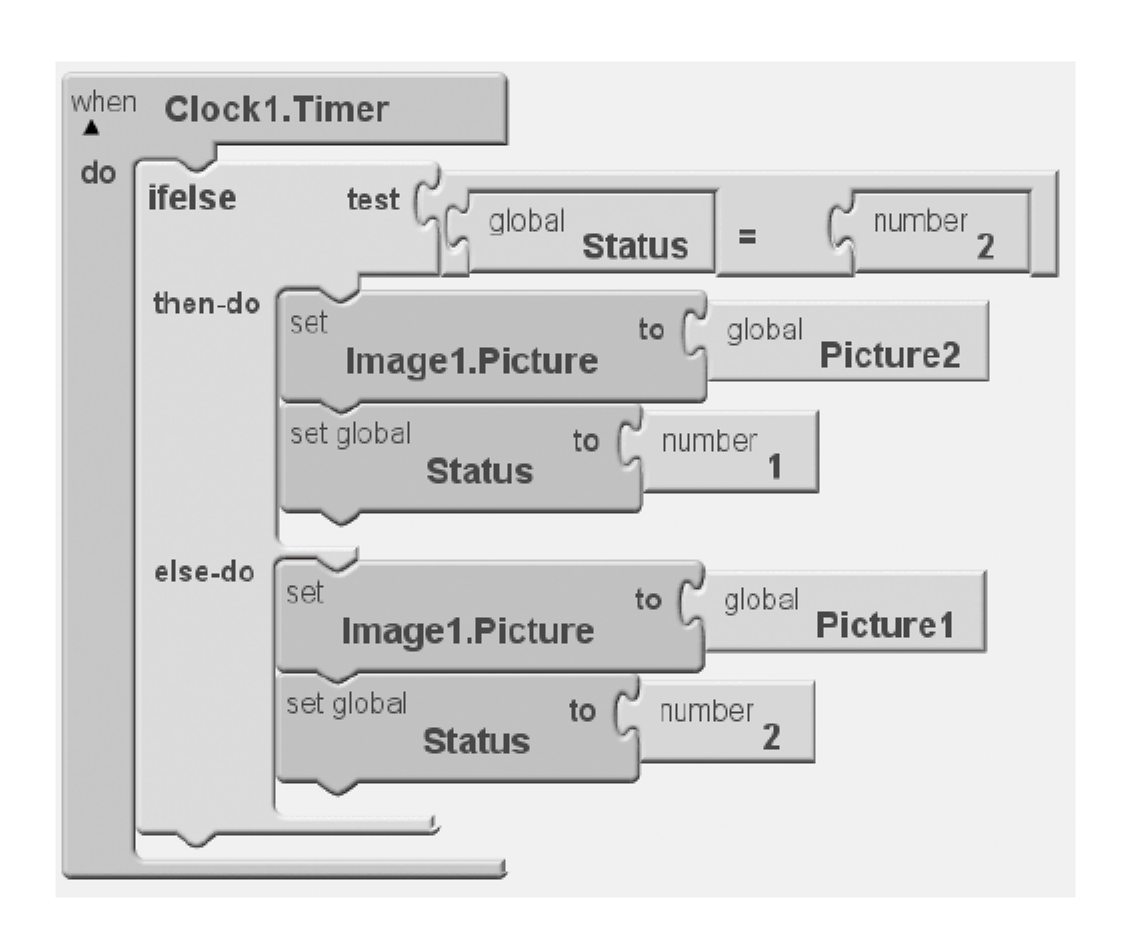


图 5.74 时钟 Clock1 的 Timer 事件

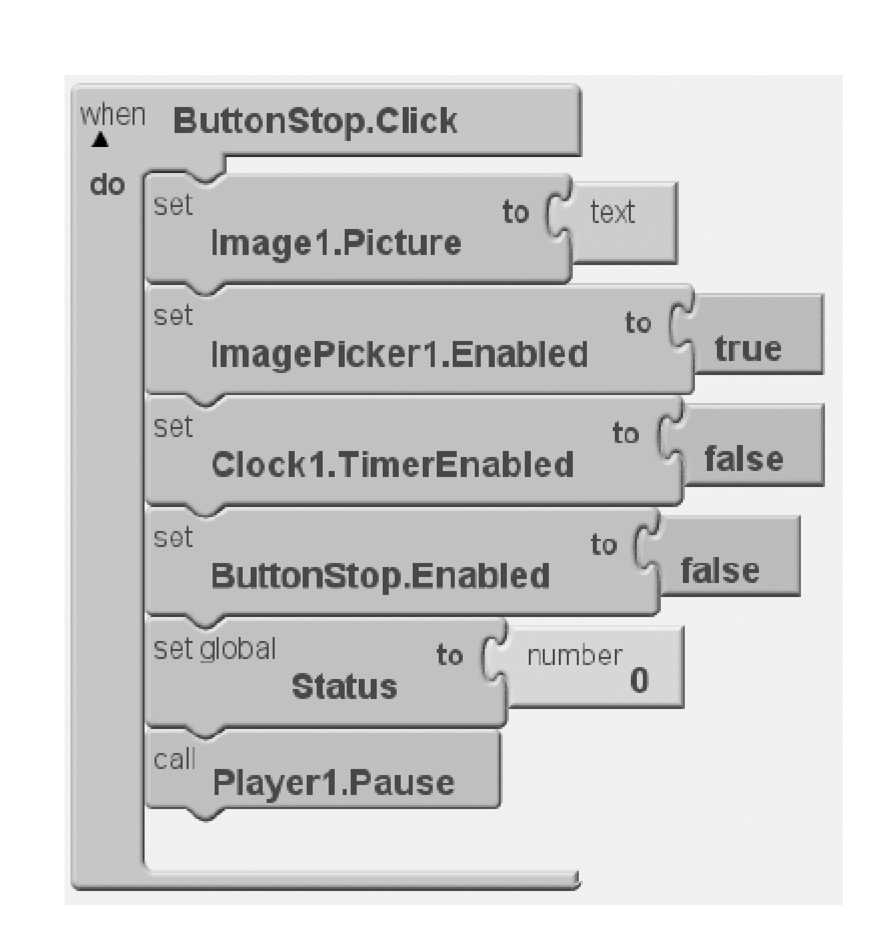


图 5.75 ButtonStop 按钮的 Click 事件

停止音乐播放是调用 Player1 的 Pause 方法。这里没有使用 Stop 方法，因为使用 Stop 方法后会引发异常，而且音乐要在用户再次单击“启动相册”按钮时继续播放，所以使用 Pause 方法更加适合。

将图像控件 Image1 的 Picture 属性赋值为空，则可以清空图像控件上所显示的图片。

MusicGallery 示例的全部逻辑模块如图 5.76 所示。

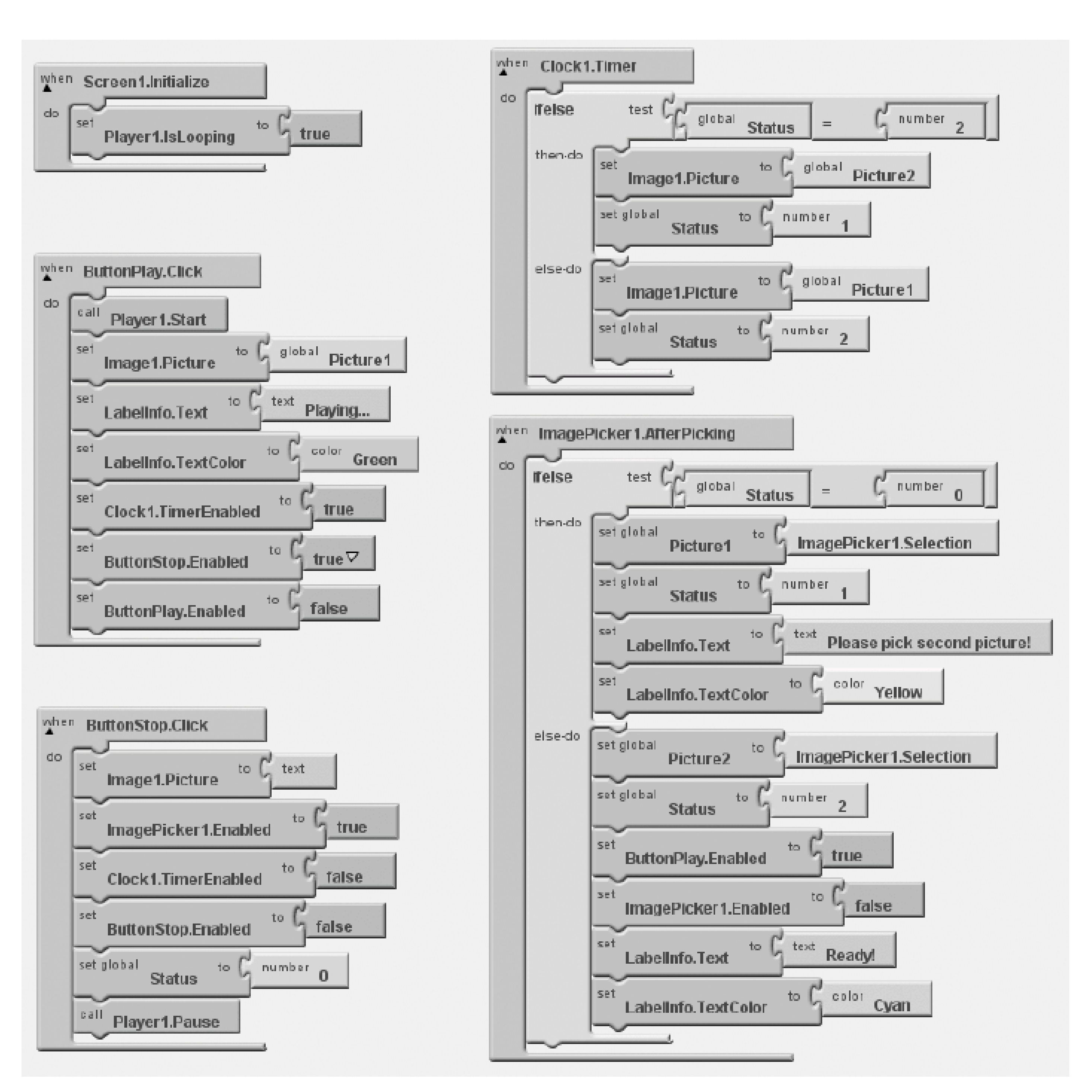


图 5.76　MusicGallery 示例的全部逻辑模块

5.5　社交控件

时至今日，手机的用途早已不仅局限于单纯的打电话，其社交功能也成为基本功能之一。App Inventor 的社交控件支持拨打电话、发送短信和选取联系人等功能，全部的社交控件如图 5.77 所示。

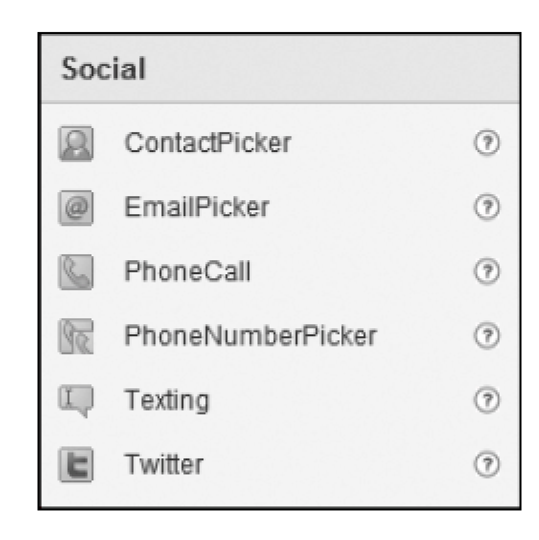


图 5.77　社交控件

需要注意的是，在 PhoneCell 和 Texting 控件的使用过程中，可能会产生一定的通话费用。并且，社交控件会从手机的联系人中获得真实信息，因此在软件开发和调试过程中，不要干扰到他人的正常生活。

在表 5.16 中给出了社交控件的具体功能说明。因为防火墙对国内出境数据的选择性过滤等原因，Twitter 控件可能暂时无法使用，因此本书中将暂不介绍 Twitter 控件的使用方法。

表 5.16　社交控件功能

控　　件	说　　明
ContactPicker	打开手机的电话簿，选取联系人信息
EmailPicker	辅助用户完成电子邮件地址输入
PhoneCell	拨打电话
PhoneNumberPicker	打开手机的电话簿，选取联系人电话号码
Texting	发送短信息
Twitter	可以与 Twitter 通信的非可视化控件

5.5.1 选取联系人

选取联系人(ContactPicker)控件的功能是让用户从手机的通讯录中获得联系人信息,这些信息包括联系人的姓名、头像和电子邮件地址。ContactPicker 在界面上的显示效果与按钮相同,如图 5.78 所示。

Text for ContactPicker1

图 5.78 联系人选取工具

当用户单击 ContactPicker 后,手机的通讯录会被打开,联系人会以列表的形式呈现,在选择目标联系人后,通讯录会自动关闭,此时联系人的信息将被保存在 ContactPicker 中。

ContactPicker 的其他外观属性,如字体、尺寸、是否可见等属性和按钮控件的性格属性相同,其所支持的专有属性如表 5.17 所示。

ContactPicker 支持 AfterPicking 事件、BeforePicking 事件、GotFocus 事件和 LostFocus 事件,事件说明如表 5.18 所示。

表 5.17 ContactPicker 属性

属 性	说 明
ContactName	联系人姓名
EmailAddress	联系人邮箱
Picture	联系人头像

表 5.18 ContactPicker 事件

事件	说 明
AfterPicking	选择后事件,在用户选择目标联系人后产生
BeforePicking	选择前事件,在用户打开通讯录,但是尚未选择目标联系人时产生
GotFocus	获取焦点事件
LostFocus	失去焦点事件

将 ContactPicker 和标签、文本框等控件结合就可以组成简单的电话簿应用程序。除此之外,还可以利用 ContactPicker 的返回结果,为其他应用提供信息,如语音留言、自动拨号和定义联系人信息等。

5.5.2 选取号码

选取号码(PhoneNumberPicker)可以获取手机通讯录中的联系人信息,这些信息包括联系人的姓名、头像、电子邮件地址和电话号码。选取号码控件如图 5.79 所示。

Text for PhoneNumberPicker1

图 5.79 选取号码控件

选取号码控件的显示效果、属性、事件和工作方式,均与 ContactPicker 相同,区别在于选取号码控件将多获取一项数据:电话号码。

二者在通讯录中选择联系人的界面也稍有区别,ContactPicker 显示的是联系人的姓名列表,而 PhoneNumberPicker 使用的则是“姓名+电话”列表,如图 5.80 所示。

5.5.3 邮件地址工具

邮件地址工具(EmailPicker)在用户输入联系人的电子邮件地址时,提供自动完成邮



图 5.80　选取信息的区别

件地址输入的功能。通常，邮件地址工具与按钮控件同时使用，在用户单击按钮控件后，自动完成邮件地址的输入和显示。

邮件地址工具在界面上的显示效果类似于文本框，没有控件方法，仅有获取焦点事件和失去焦点事件。除 Text 和 Hint 属性之外，其余用于设置外观效果等的属性和文本框控件类似，这里不再重复叙述。邮件地址工具在界面上的显示效果如图 5.81 所示。

5.5.4 拨号

拨号控件(PhoneCell)是一个非可视化控件，用于向指定的电话号码拨打电话，如图 5.82 所示。

拨号控件只有一个属性 PhoneNumber 和一个方法 MakePhoneCall。

PhoneNumber 属性中保存着目标电话号码，此属性可以在界面编辑器中设定，也可以在模块编辑器中定义和修改。

MakePhoneCall 方法（图 5.83）会调用手机的拨号界面，然后自动拨打 PhoneNumber 属性中的电话号码。如果 PhoneNumber 属性中没有设定任何电话号码，MakePhoneCall 方法将不会被执行。



图 5.81　邮件地址工具



图 5.82　拨号控件



图 5.83　MakePhoneCall 方法

拨号控件通常与选取号码控件结合使用。其通常的使用方法是将选取号码控件中获得的电话号码，赋值给拨号控件的 PhoneNumber 属性，然后再调用拨号控件的 MakePhoneCall 方法拨打电话。

5.5.5 短信息

短信息控件(Texting)主要用来发送和接收短信息。短信息控件也是非可视化控件,在界面编辑器中的效果如图5.84所示。

短信息控件的主要属性如表5.19所示。其中,GoogleVoiceEnabled属性表示是否启动GoogleVoice功能。如果用户有GoogleVoice账号,短信息将利用WiFi网络,通过GoogleVoice进行发送。PhoneNumber属性表示发送短信的目标电话号码,形式是一组数字,如18600001111,但不能包含标点符号和空格。

Non-visible components

Texting1

图5.84 短信息控件

表5.19 短信息控件的属性

属　　性	说　　明
GoogleVoiceEnabled	是否允许使用谷歌语音功能
Message	发送的短信息内容
PhoneNumber	目标电话号码
ReceivingEnabled	是否允许应用接收短信息

ReceivingEnabled属性有三个可选值:数值1,表示忽略接收短信息功能;数值2,表示程序运行时才会接收短信;数值3,表示后台接收短信,即使程序已经退出,仍会接收短信。后台接收短信后,手机就会在通知栏内显示一条通知,选择此通知后,程序会自动被切换到前台。

短信息控件支持MessageReceived事件,在接收到短信息后产生,可以提取短信发送方的号码和短信息内容,如图5.85所示。

短信息控件支持SendMessage方法(图5.86),用来发送短信息,目标电话号码在PhoneNumber属性中,发送内容在Message属性中。



图5.85 MessageReceived事件



图5.86 SendMessage方法

接下来,通过PhoneBook示例实际应用社交控件,制作一个简单的通讯录应用。

PhoneBook示例利用选取号码控件在电话簿中选取目标联系人的电话号码,然后使用短信息控件和拨号控件对此号码发送短信息或拨打电话。需要注意的是,由于此示例使用了手机间的通信功能,所以需要使用实体手机进行调试。

通讯录是手机中使用最为频繁的软件,一般按照联系人名字的字母顺序排列,可以显示联系人的姓名、头像、手机号和电子邮件地址等信息,如图5.87所示。用户选择联系人后,可以拨打电话或发送短信。通讯录的数据信息保存在手机的数据库中,一般通

过共享机制，允许其他软件读取数据库中的联系人信息。

PhoneBook 示例是一个简化的通讯录，通过读取数据库中的联系人信息，可以显示联系人的姓名、电话和头像，并支持拨打电话、发送接收短信息的功能。PhoneBook 示例的运行界面如图 5.88 所示。



图 5.87　手机通信录



图 5.88　PhoneBook 示例运行界面

单击“选取联系人”按钮，将弹出手机内置的联系人选择界面，如图 5.89 所示，供用户选择目标联系人。

选择“李雷”这个联系人后，PhoneBook 示例运行界面将跳转回启动时的主页面，此时“李雷”的头像、姓名和电话已经显示在界面上，如图 5.90 所示。



图 5.89　手机内置的联系人选择界面



图 5.90　包含“李雷”信息的界面

此时单击“打电话”按钮，将直接呼叫李雷的电话号码，也可单击“发短信”按钮，将文本框中的内容发送到李雷的电话号码上。需要注意的是，拨打电话和发送短信会产生一定的通信费用。

在 PhoneBook 示例的界面示意图中，包含两个非可视化控件 Texting1 和

PhoneCall1，如图 5.91 所示。AfterPicking 事件如图 5.92 所示。

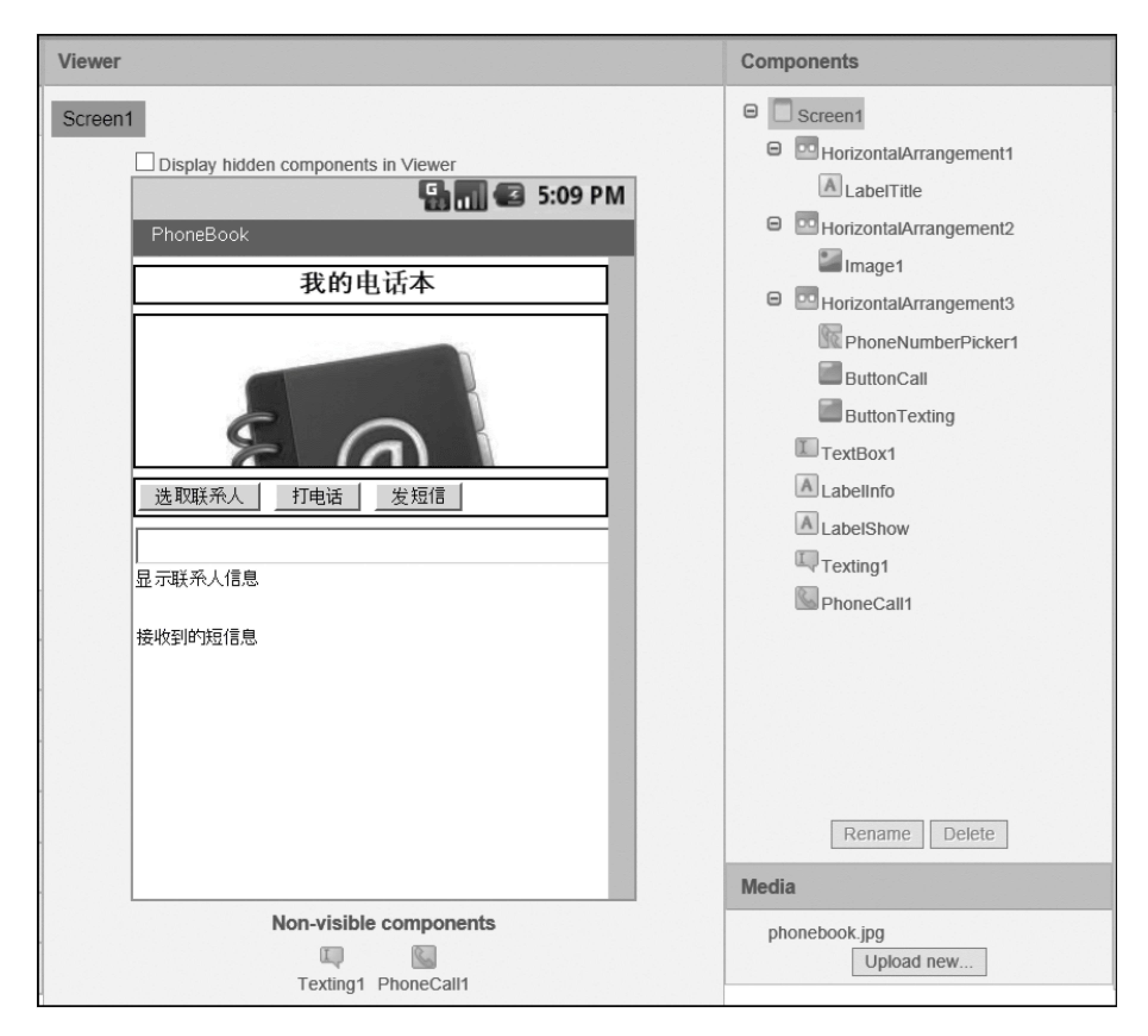


图 5.91　PhoneBook 示例界面示意图

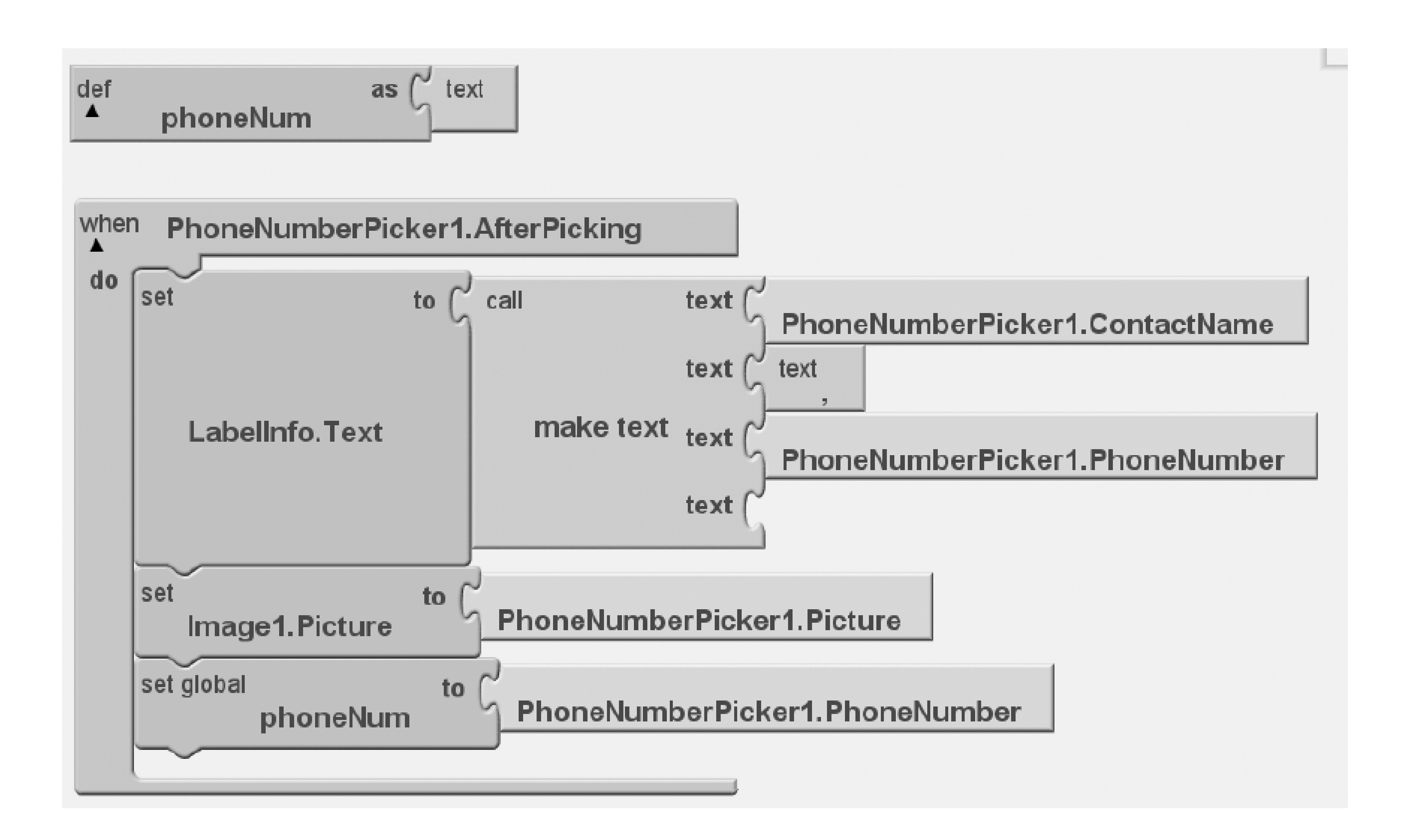


图 5.92　AfterPicking 事件

PhoneNumberPicker 控件被单击后，会自动出现“联系人选择界面”，因此无须处理该控件的单击事件。但一般都会在 AfterPicking 事件中，保存或者显示目标联系人的相关信息。PhoneBook 示例中，将联系人的头像 Picture 赋值给图像控件 Image1 的 Picture 属性，并将电话号码 PhoneNumber 赋值给全局变量 phoneNum，供拨打电话和发送短信功能使用。

在拨打电话前,要先设置 PhoneCall1 控件的 PhoneNumber 属性。将全集变量 phoneNum 中保存的电话号码赋值给 PhoneNumber 属性后,就可以调用 MakePhoneCall 方法拨打电话了,如图 5.93 所示。

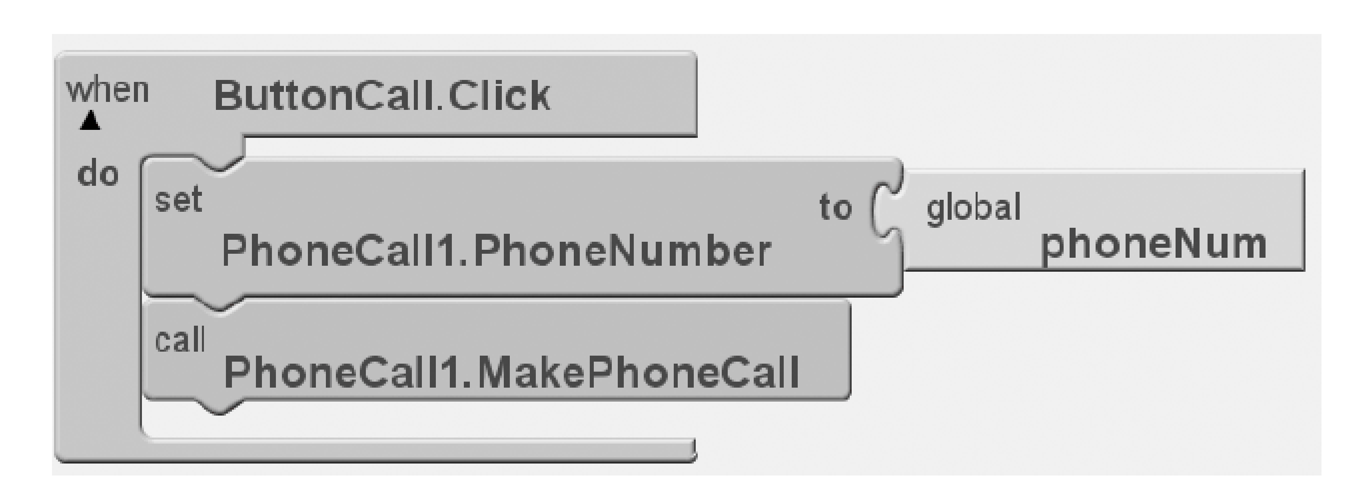


图 5.93 拨打电话功能

发送短信功能中有一个小技巧,就是避免用户发送空短信。首先判断短信内容是否为空,如果短信内容不为空,才可以进行实质的短信发送过程。发送短信前,要设置 Texting1 控件的 PhoneNumber 属性和 Message 属性,然后调用 SendMessage 方法发送短信,如图 5.94 所示。

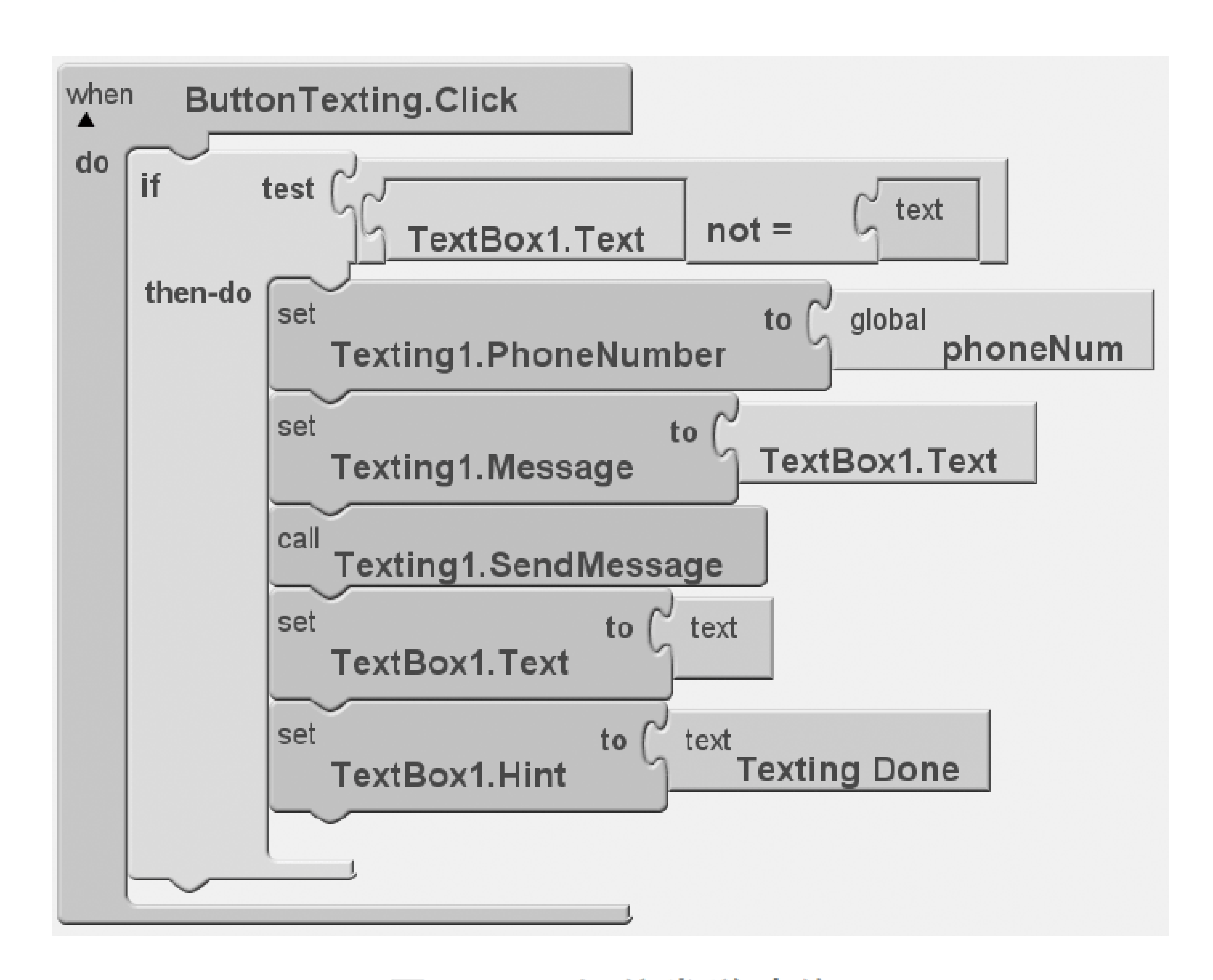


图 5.94 短信发送功能

短信接收功能非常简单,如图 5.95 所示,在 Texting1 的 MessageReceived 事件中,

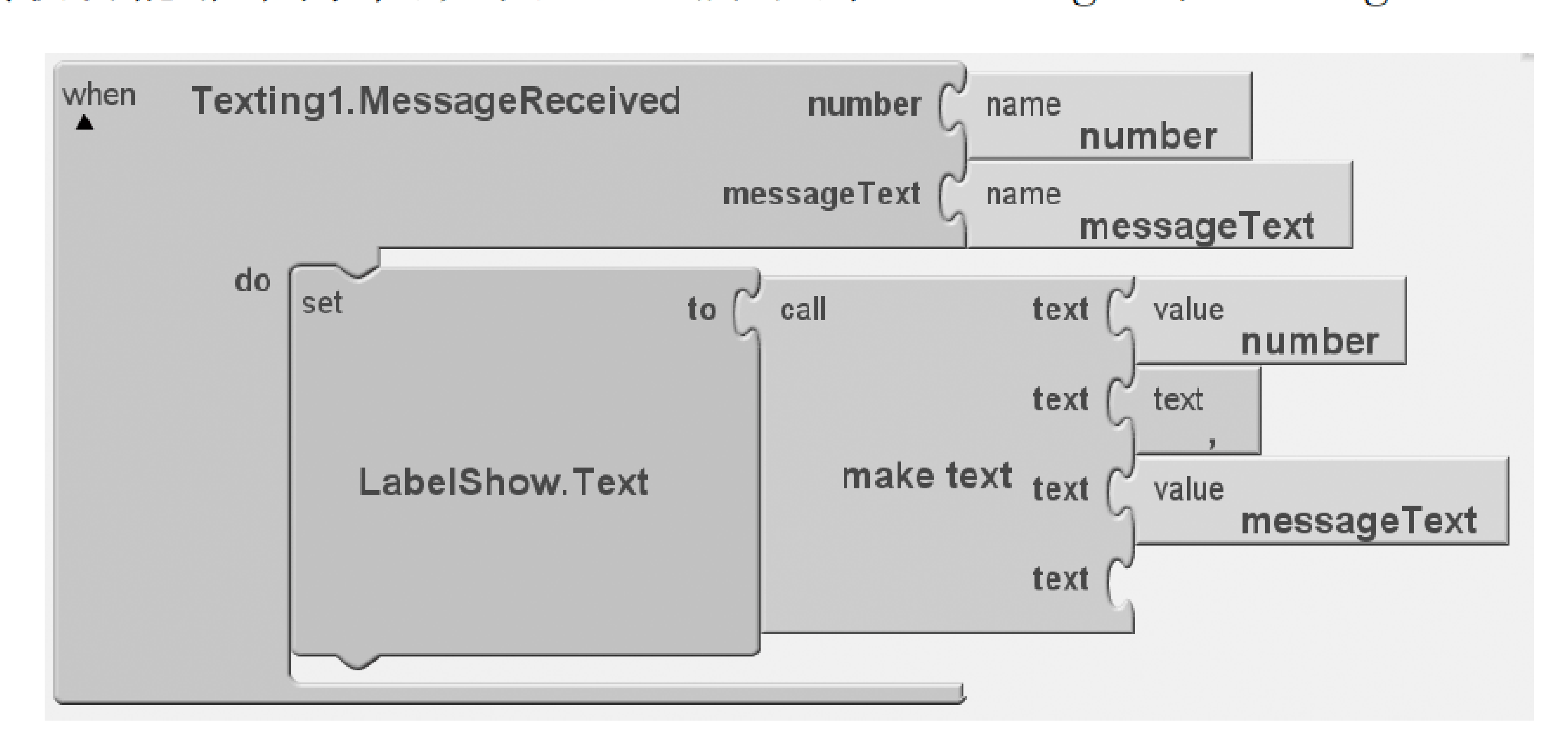


图 5.95 短信接收功能

将事件传递的 number(短信发送者电话号码)和 messageText(短信内容)一同显示在标签控件 LabelShow 上即可。其中,将 Texting1 的 ReceivingEnabled 参数设置为 2 (foreground),表示仅在软件运行期间接收短信息。

PhoneBook 示例全部逻辑模块如图 5.96 所示。

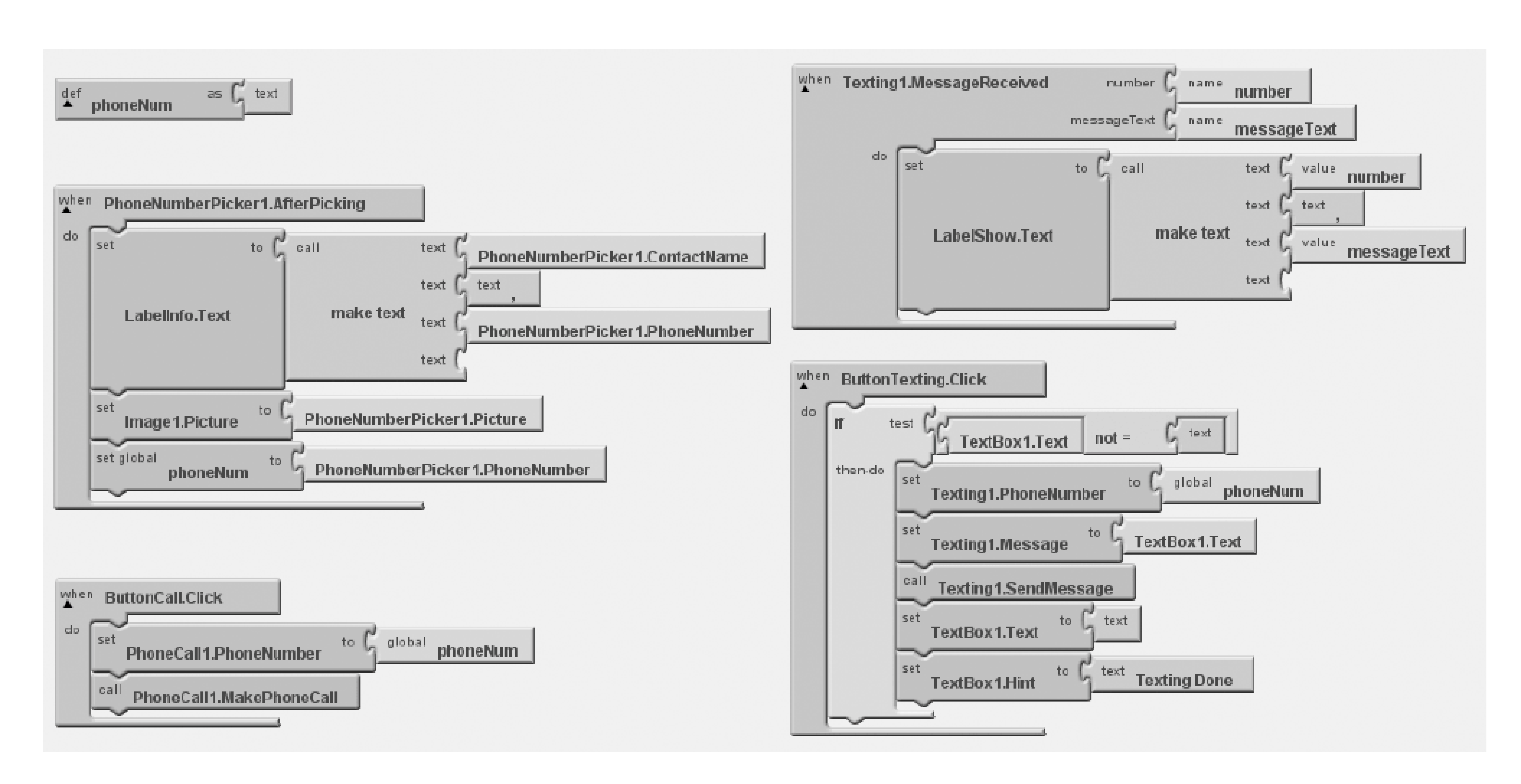


图 5.96 PhoneBook 示例全部逻辑模块

习 题

1. 音频播放器控件和音效控件在使用场合和使用方式上有何区别?

2. 选取联系人控件的功能能否被选取号码控件替代? 为什么?

3. 完成一个简单的应用,界面效果如图 5.97 所示,要求通过列表选项在不同的曲目间进行选择(曲目数量及内容自拟),并通过按钮控制播放和停止。



图 5.97 音乐播放器界面效果

第 6 章



动画与游戏

动画效果是游戏中不可缺少的内容,本章主要介绍画布(Canvas)和精灵(ImageSprite)的使用方法,以及一些高级的动画功能,如边缘检测、碰撞处理和精灵操纵等。

本章学习目标:

- 掌握画布的使用方法;
- 理解画布的坐标系统;
- 掌握精灵的使用方法;
- 掌握球(Ball)的使用方法;
- 了解边缘检测和碰撞处理的原理。

6.1 画 布

6.1.1 画布介绍

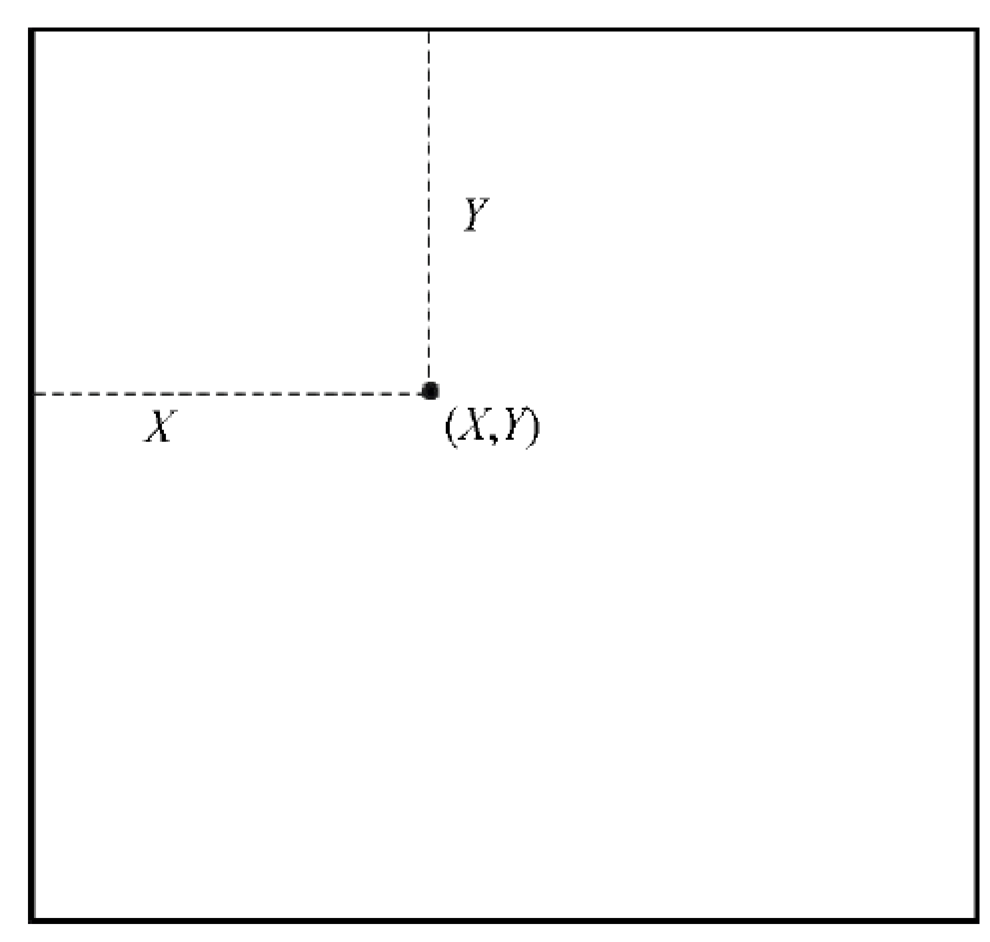
画布是一种可在其上绘制图像的控件,初始的画布像是一张空白的幻灯片,没有任何内容,用户可以在画布上绘制各种图形,例如线条、点、矩形或圆形,也可以在画布上加载图片作为画布的背景,或是在画布上显示文字。画布是常用的控件,除了作为绘制图形的承载体以外,还经常作为游戏的背景画面。

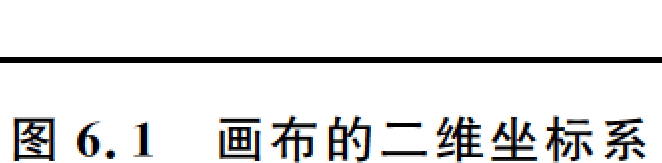
画布是一个具有触控感应的二维平面图板,采用经典的二维坐标系,坐标的原点在画布的左上角。下面以图 6.1 中的坐标点(X,Y)说明画布的坐标系,X 表示坐标点距离左侧边缘的距离,Y 表示坐标点距离上边缘的距离,且 X 和 Y 都取正值。

6.1.2 画布使用

在基本控件库中可以找到画布(Canvas),将画布拖曳到界面设计区后,画布仅显示为一个小图标,如图 6.2 所示。如果要让画布填充满整个屏幕,只需手动设置画布的宽度(Width)和高度(Height)。

除了可以使用宽度和高度属性控制画布的大小以外,画布还支持更改背景颜色(BackgroundColor)和更改背景图片(BackgroundImage)、设置画笔的颜色(PaintColor)




图 6.1 画布的二维坐标系

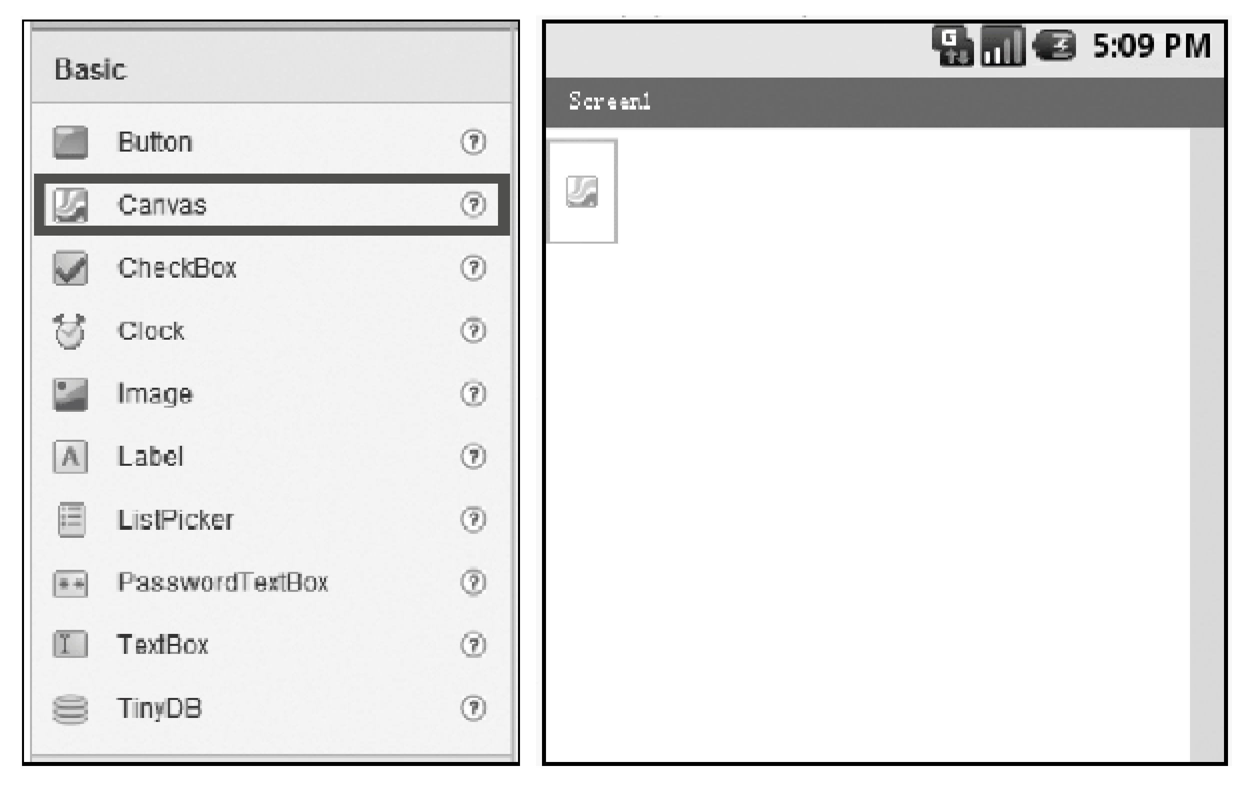


图 6.2 画布控件

和宽度(LineWidth),以及画布是否可见(Visible)等属性。画布的全部属性如图 6.3 所示,属性说明如表 6.1 所示。

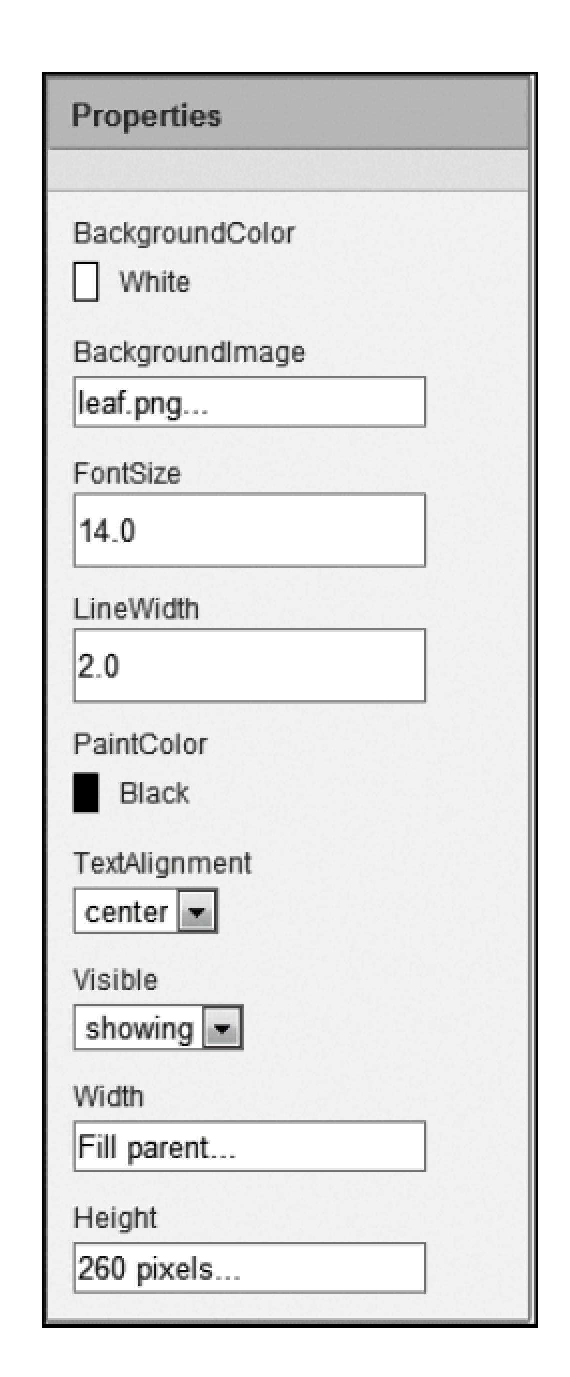


图 6.3 画布属性

表 6.1 画布属性

属　　性	说　　明
BackgroundColor	背景颜色
BackgroundImage	背景图片
FondSize	字体大小
LineWidth	画笔宽度
PaintColor	画笔颜色
TextAlignment	文字排列方式
Visible	画布控件是否可见
Width	画布宽度
Height	画布高度

画布支持的事件有拖曳事件(Dragged)、触碰事件(Touched)、快速划动事件(Flung)、触碰按下事件(TouchDown)和触碰抬起事件(TouchUp),如表 6.2 所示。

表 6.2 画布事件

事　件	说　　明
Dragged	拖曳事件
Flung	快速划动事件
TouchDown	触碰按下事件
TouchUp	触碰抬起事件
Touched	触碰事件,由触碰按下动作和触碰抬起动作组成

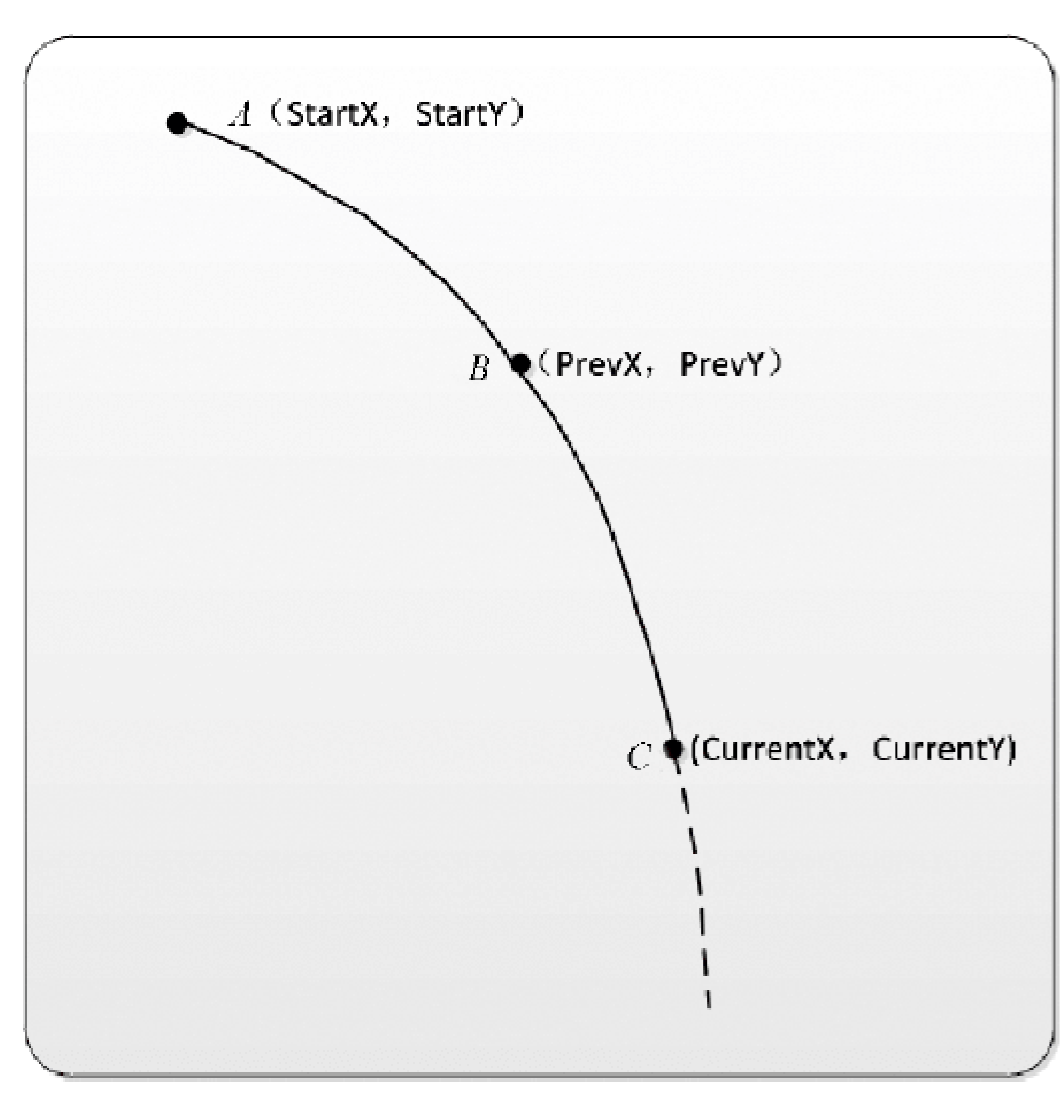


图 6.4 画布中二维坐标与事件参数

下面以图 6.4 的场景为例，来说明画布事件的响应规律。如果手指按住 A 点，此时将产生 TouchDown 事件，手指从 A 点抬起，将产生 TouchUp 事件和 Touched 事件。

如果手指从 A 点按下，缓慢地沿弧线，从 B 点滑动到 C 点，然后抬起手指。当手指接触到 A 点时，将产生 TouchDown 事件，在手指从 A 到 B 再到 C 的移动过程中，将多次产生 Dragged 事件，在手指离开 C 点时，将产生 TouchUp 事件。可见，无论手指在屏幕上如何移动，TouchDown 和 TouchUp 事件只在手指触碰到屏幕和离开屏幕时产生。如果手指的触碰点和抬起点是同一点，将引发 Touched 事件，而如果手指在屏幕上移动了，则不会引发 Touched 事件。TouchDown 事件、TouchUp 事件和 Touched 事件可以为用户提供触碰点的坐标，有一个需要注意的事情，TouchUp 提供的坐标点，是手指按下时的坐标，而不是手指抬起时的坐标。

Dragged 事件在手指移动过程中持续产生，主要用来跟踪手指的移动轨迹。Dragged 事件提供三个坐标，分别是移动开始坐标点(StartX 和 StartY)、事件产生时的当前坐标点(CurrentX 和 CurrentY)、前一个事件产生时的坐标点(PrevX 和 PrevY)。以图 6.4 的场景为例，如果从 A 点到 C 点的过程中，只在 B 点和 C 点产生了 Dragged 事件，则在 C 点产生的事件中，移动开始节点是 A，当前节点是 C，前一个事件产生是在 B 点。

Flung 事件只有手指在屏幕上快速划动的时候才会产生。同样是从 A 经过 B 到 C 的过程，如果手指在屏幕上移动得足够快，则当手指在 C 点抬起时，会产生 Flung 事件。Flung 事件中会提供划动开始节点的坐标、划动方向、速度、速度在 X 轴和 Y 轴的分量。

如图 6.5 所示为逻辑编辑器中的画布事件。

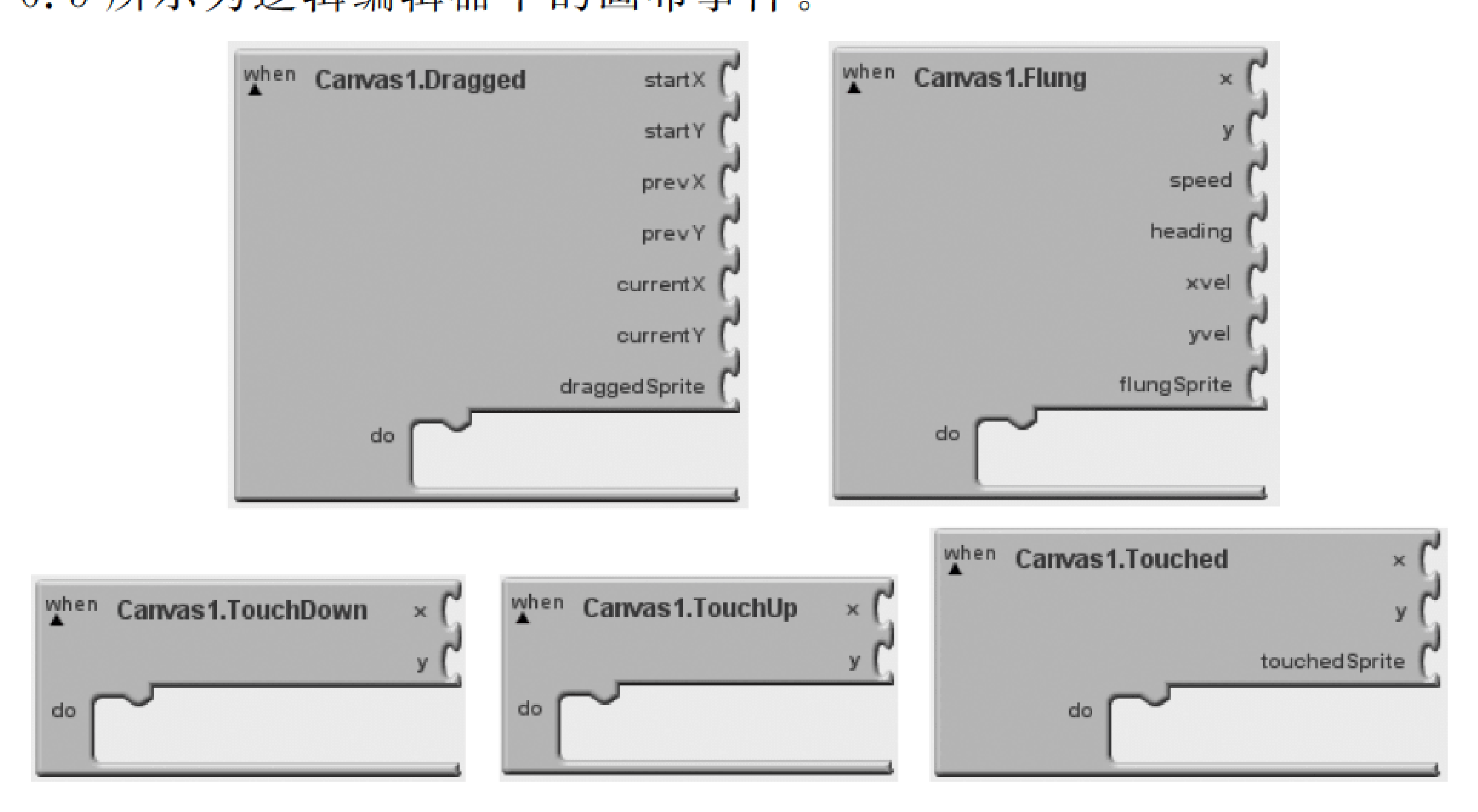


图 6.5 逻辑编辑器中的画布事件

下面用 CanvasEvent 示例说明画布的事件响应的时机和提供的数据。图 6.6 分别是在界面编辑器中的界面和手机运行时的界面。

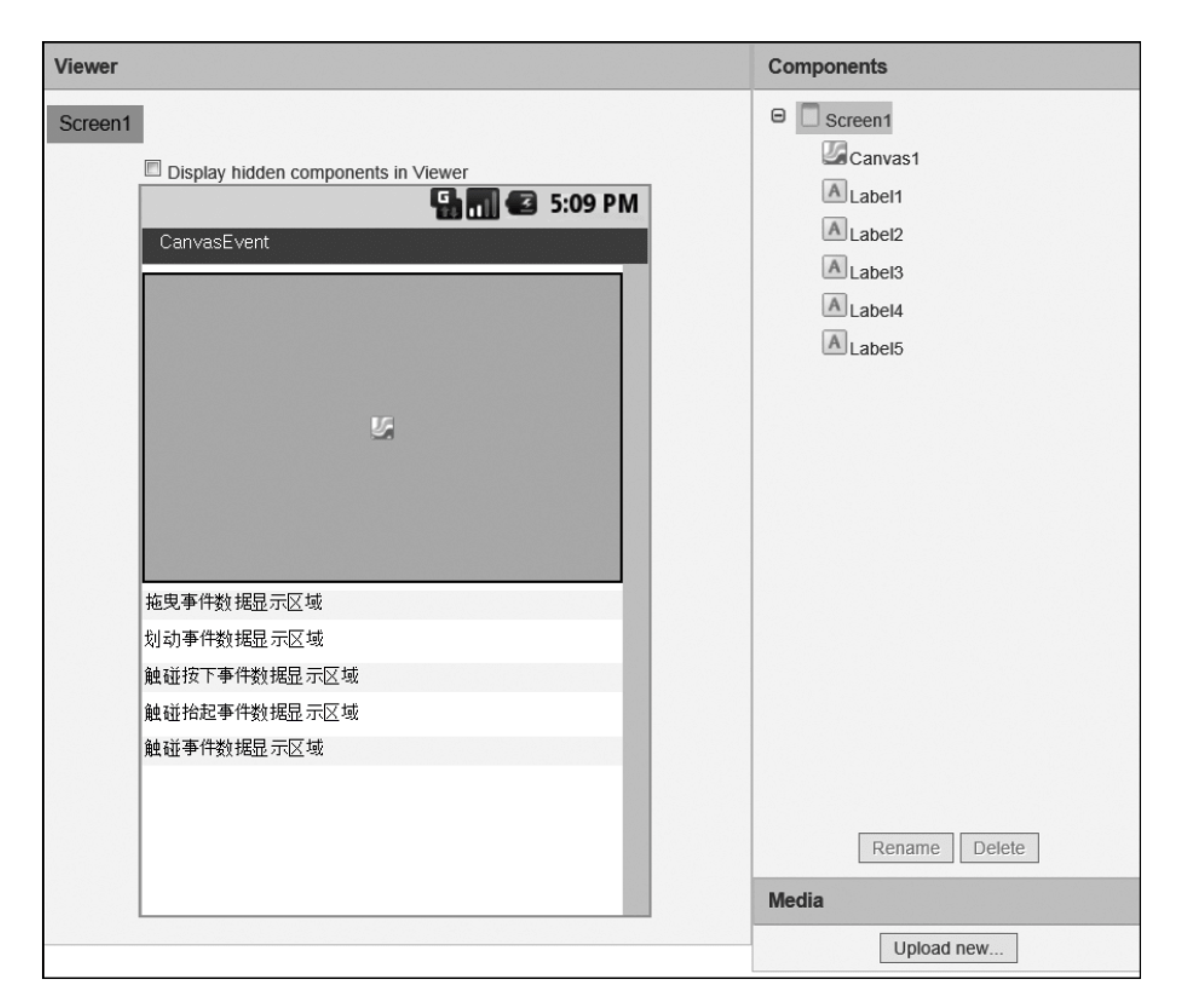


图 6.6　界面编辑器中的 CanvasEvent 示例

CanvasEvent 示例(图 6.7)中上方是画布控件,下方是各种事件数据的显示区域。在上方的画布上进行触碰操作或滑动操作,事件的响应结果便会显示在界面下部。

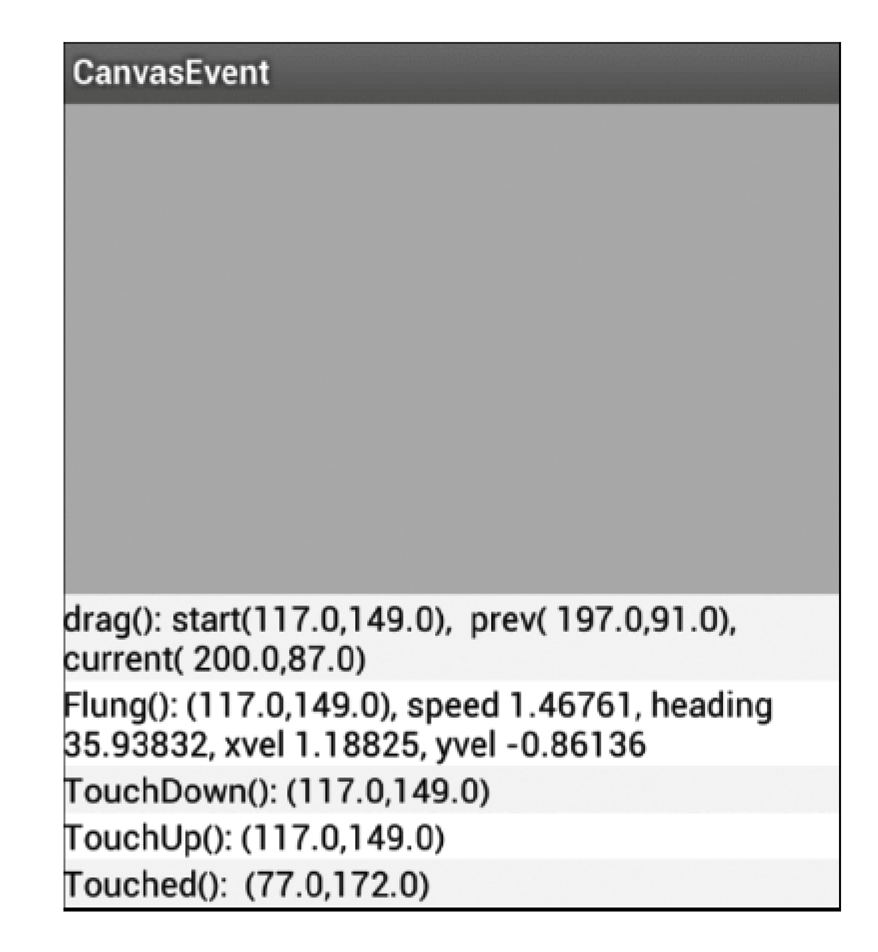


图 6.7　CanvasEvent 示例运行界面

CanvasEvent 示例的逻辑部分也较为简单。在每个事件的响应函数中,将事件所可以提供的数据,格式化为一个字符串显示在界面固定位置上。如图 6.8 所示为 Flung 事件和 Dragged 事件处理函数,图 6.9 为 TouchDown 事件和 TouchUp 事件处理函数,图 6.10 为 Touched 事件处理函数。

通过 CanvasEvent 示例,用户可以充分理解画布所支持的事件,以及事件的使用方法。下面将介绍画布所支持的方法,例如清空画布、绘制图形、画布保存等。画布所支持的全部方法如表 6.3 所示。

画布所支持的方法,如 Clear 和 Save 是不需要用户提供任何参数的,但绝大多数方法都需要用户提供坐标作为基本参数,如 DrawPoint 和 GetPixelColor。图 6.11 给出了画布所有方法需要用户提供的参数。

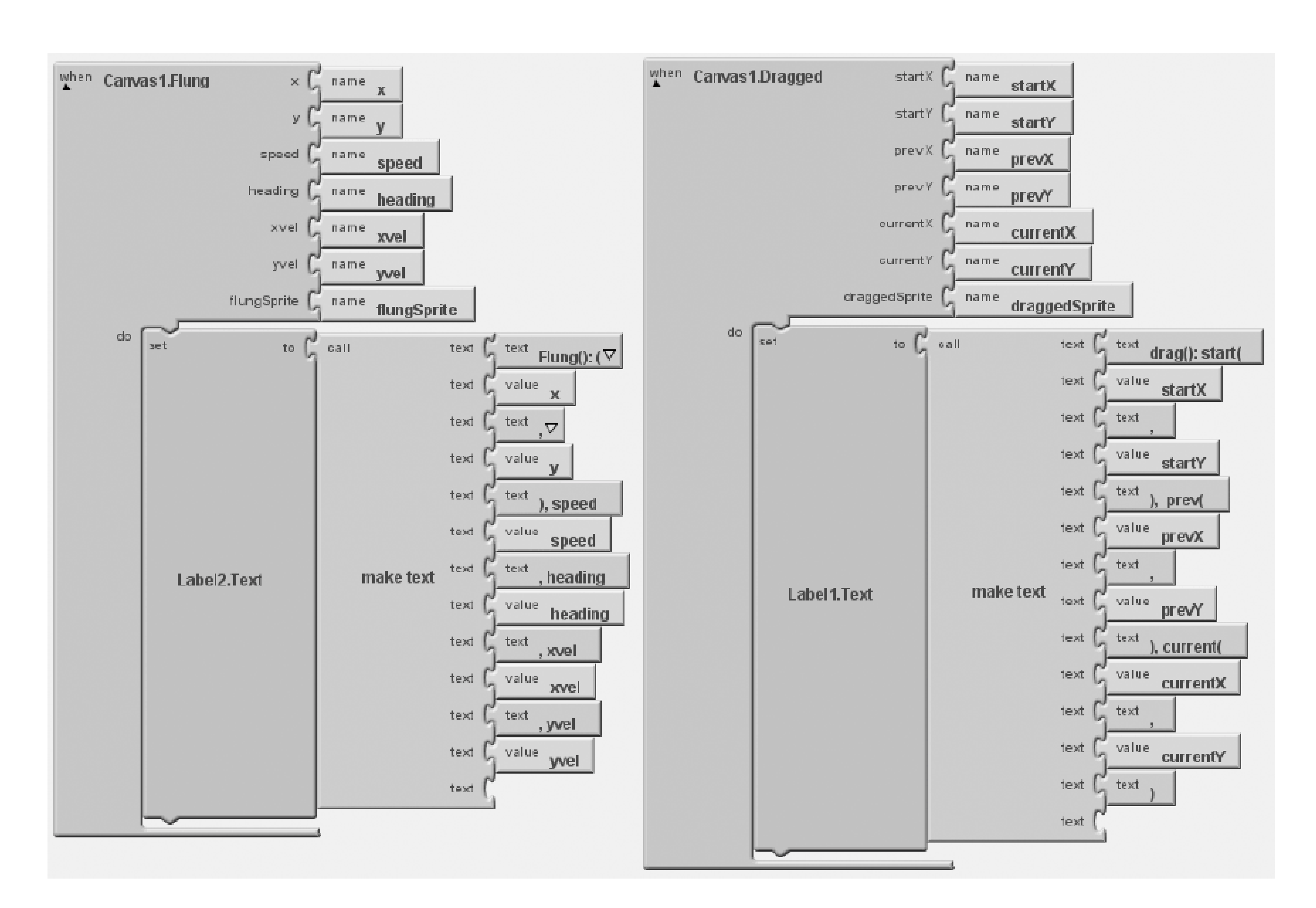


图 6.8　Flung 事件和 Dragged 事件处理函数

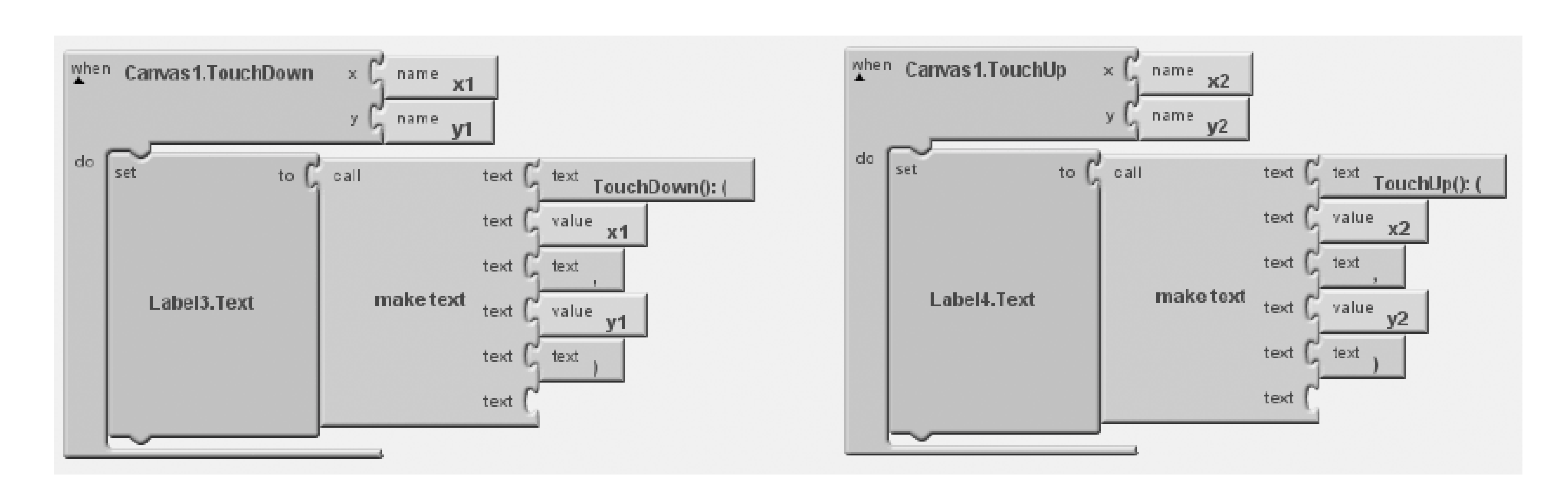


图 6.9　TouchDown 事件和 TouchUp 事件处理函数

表 6.3　画布控件方法说明

方　　法	说　　明
Clear	清除画布
DrawCircle	绘制圆形图形
DrawLine	绘制线条
DrawPoint	绘制圆点
DrawText	绘制文字
DrawTextAtAngle	以一定角度绘制文字
GetBackgroundPixelColor	获取背景图片指定像素的颜色
GetPixelColor	获取图片指定像素的颜色
Save	画布保存
SaveAs	画布另存为
SetBackgroundPixelColor	设置背景图片特定像素的颜色

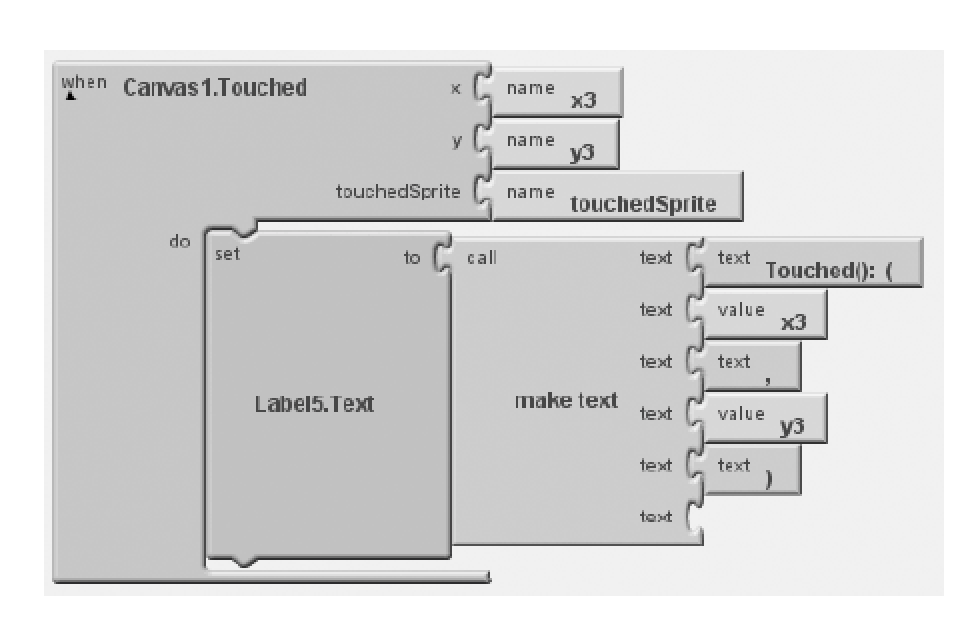


图 6.10　Touched 事件处理函数

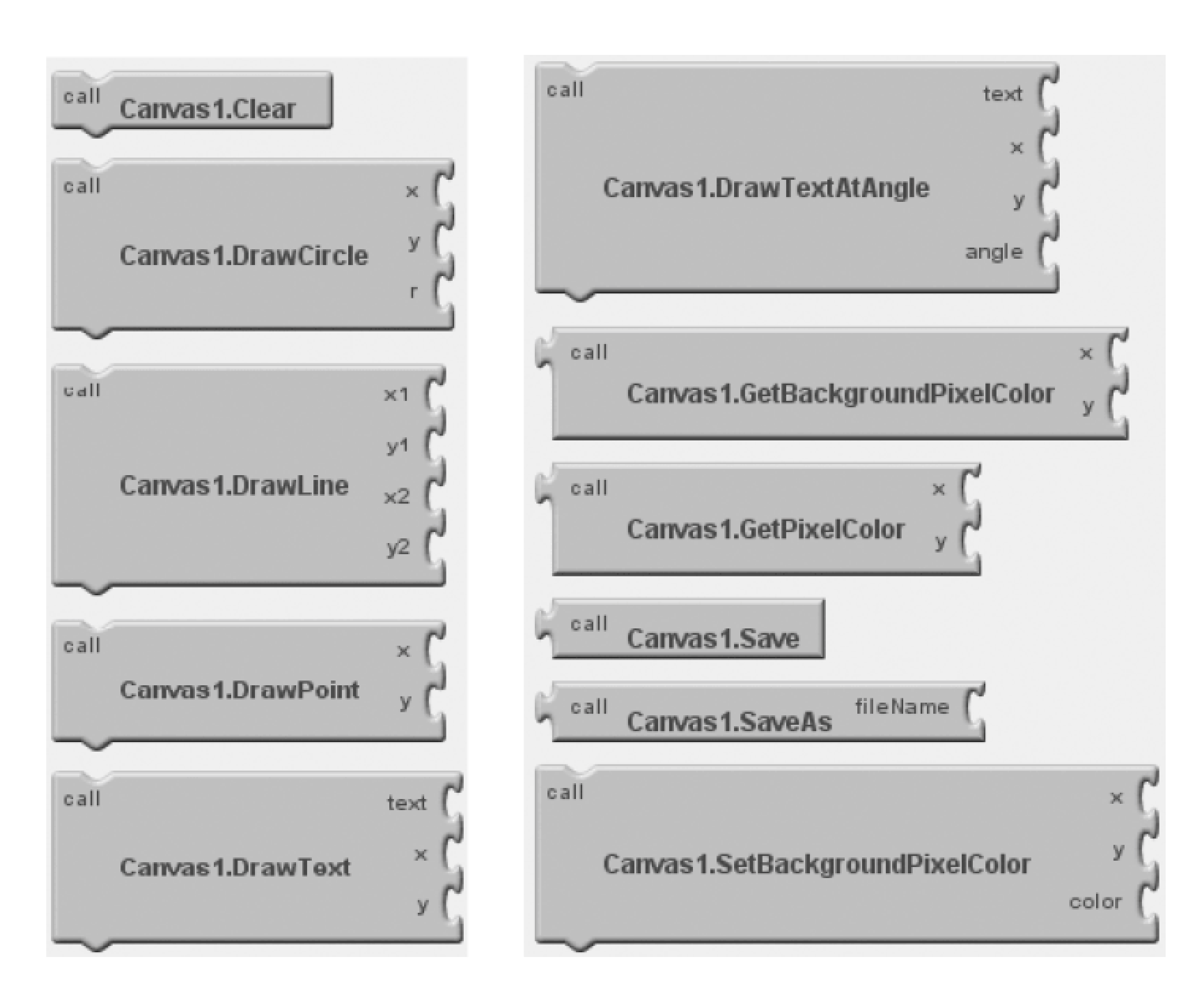


图 6.11 画布的方法

Clear 方法可以清空画布元素，如果画布上已设置图片，设置的图片也会被清除。DrawCircle 在画布上(x，y)点绘制一个半径为 r 的圆形。DrawLine 会在画布上从($x1$，$y1$)点到($x2$，$y2$)点绘制一条直线。DrawPoint 在画布(x，y)点上绘制一个圆点。DrawText 会在(x，y)点上绘制文字 Text。DrawTextAtAngle 会在(x，y)点以角度 angle 绘制文字 Text。GetBackgroundPixelColor 获取背景图片上(x，y)点的颜色。GetPixelColor 获取前景上(x，y)点的颜色。SetBackgroundPixelColor 设置背景图片上(x，y)点的颜色为 color。

Save 将画布图像存储到外部存储器的默认路径中，并返回存储的路径和文件名，若存储发生错误，则会返回错误信息。SaveAs 可以将画布图像存储到外部存储器中，并指定文件名为 filename，文件名的后缀必须为“jpeg”、“jpg”或“png”。

6.1.3 相机与加速度传感器

在介绍“画图板”示例之前，先来说明一下如何使用手机的相机和加速度传感器。

相机控件是一个非可视化控件，可利用手机的镜头进行拍照。相机控件在媒体控件区(Media)可以找到，如图 6.12 所示。

相机控件只支持一种方法 TakePicture，如图 6.13 所示，此方法被调用时，手机将进行拍照。该方法结束后会引发 AfterPicture 事件。

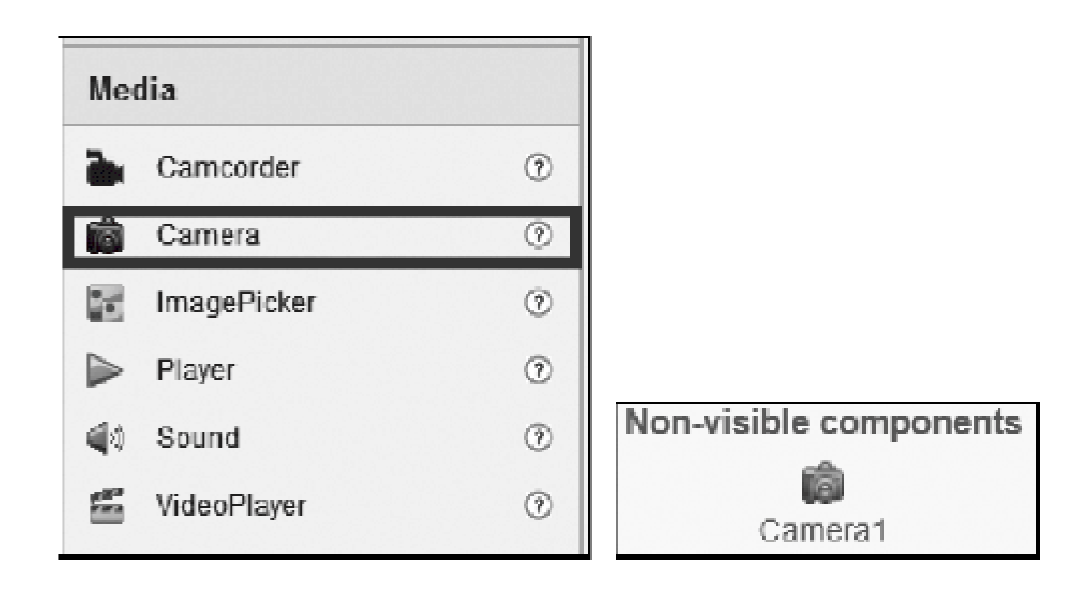


图 6.12 相机控件

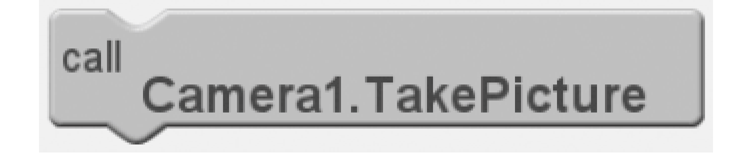


图 6.13 相机控件的 TakePicture 方法

AfterPicture 事件在拍照完成后产生，其中 image 是手机中用来存放拍摄照片的路径信息，如图 6.14 所示。

加速传感器控件用来检测手机加速度大小，可在三个方向测量手机晃动时的加速度，测量单位为米/秒²（m/s^2）。加速传感器控件在传感器控件区（Sensors）可以找到，如图 6.15 所示。




图 6.14　相机的 AfterPicture 事件

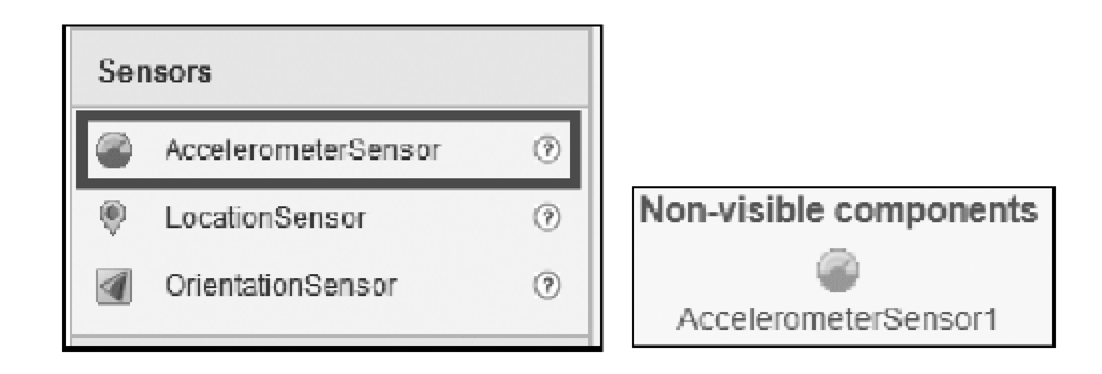


图 6.15　加速传感器控件

加速传感器控件支持 X 加速度、Y 加速度、Z 加速度，每个加速度数值都有正值和负值，这类似于坐标系中数轴上的数值。其中，设备处于水平位置向右倾斜，即设备左侧抬高时，X 加速度数值为正值；反之，设备向左倾斜，即右侧抬高时，X 加速度数值为负值。手机处于水平位置，当下部抬起时 Y 加速度为正值；反之，手机上部抬起时 Y 加速度为负值。手机处于水平位置，当屏幕朝上时 Z 加速度为正值；屏幕朝下时 Z 加速度为负值。此外，当手机屏幕朝上且水平放置时，Z 加速度约为 $9.8m/s^2$。

加速传感器控件还支持 MinimumInterval、Available 和 Enabled 属性，属性的具体含义如表 6.4 所示。

表 6.4　加速传感器控件的属性

属　性	说　　明	属　性	说　　明
Available	手机是否具有加速感应器件	YAccel	垂直加速度
Enabled	加速感应器控件是否可用	ZAccel	竖直方向加速度
XAccel	水平加速度	MinimumInterval	手机晃动的最小间隔

加速传感器控件支持的事件有 AccelerationChanged 和 Shaking，如图 6.16 所示。AccelerationChanged 事件在加速传感器的加速度改变时调用，并根据加速传感器的变化返回 X、Y、Z 加速度值，可以在三个方向上确定手机晃动时的加速度大小。Shaking 事件在手机摇晃时会被多次调用。

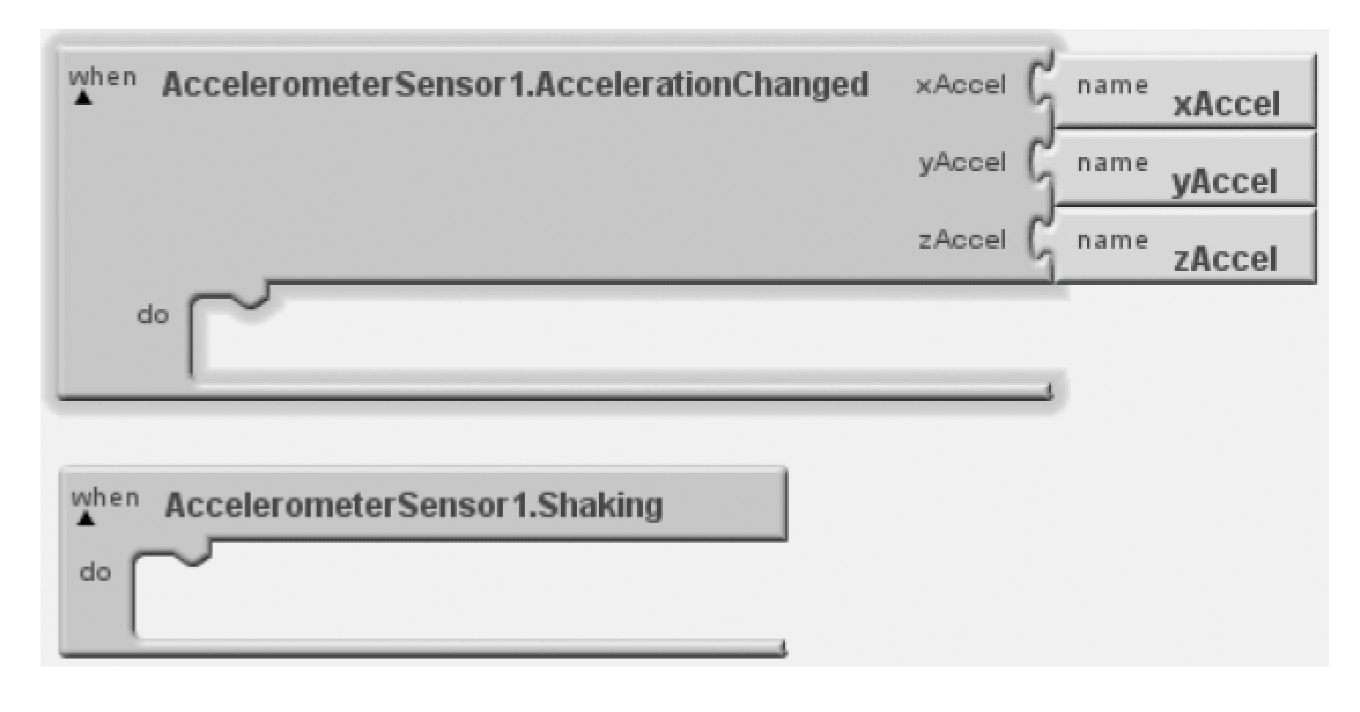


图 6.16　加速传感器控件的事件

6.1.4 示例——画图板

前面的内容详细地介绍了画布的使用方法，下面介绍一个更为复杂的示例 PaintPic。在 PaintPic 示例中，除了用到画布控件之外，还使用了相机控件（Camera）和加速传感器控件（AccelerometerSensor）。



图 6.17　PaintPic 示例运行界面

PaintPic 示例主要实现画布功能，可以利用画笔在画布上绘制线条或者圆形图案，并可选不同的画笔颜色，可以使用"清空画布"按钮或者摇晃手机的方式清空画布，调用手机照相功能，将所拍摄的图片作为画布的背景，并可以将画布的背景和绘制的图像保存成文件。PaintPic 示例的运行界面如图 6.17 所示。

PaintPic 示例中大量使用了按钮和水平布局，并且使用了两个非可视化控件相机和加速传感器。这两个控件在界面设计图中可以找到，如图 6.18 中编辑区的最下方的 Camera1 和 AccelerometerSensor1 分别就是相机和加速传感器。虽然在界面设计图中可以看到非可视化控件，但在程序实际运行过程中是完全不可见的。

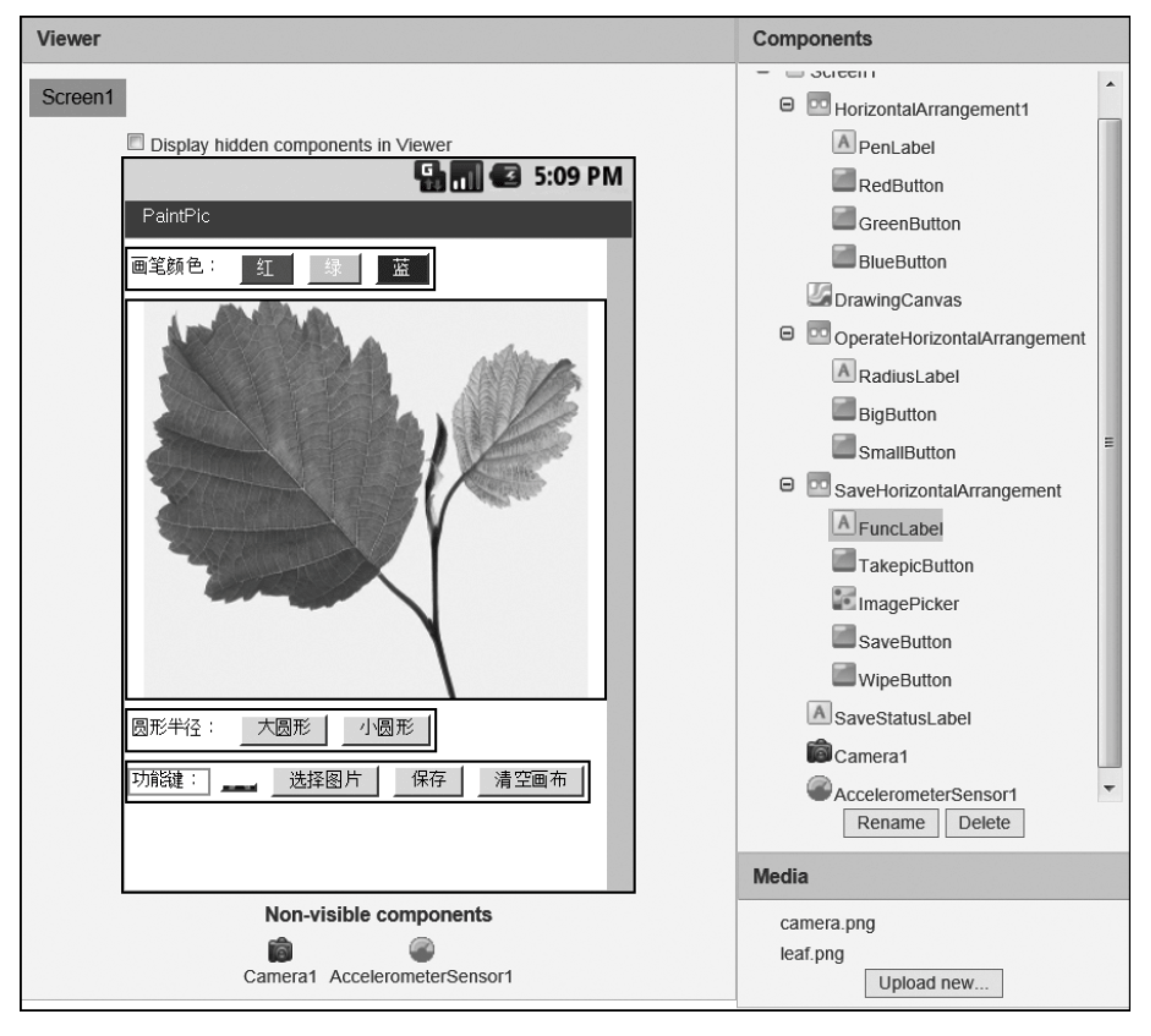


图 6.18　PaintPic 示例界面设计图

完成 PaintPic 示例的界面设计后，下面进行逻辑功能的设计。

逻辑功能设计的第一步是定义一个全局变量 paintsize，用来表示绘制圆形图案的半径。定义全局变量要从 Built-In→Definition→Variable 获取全局变量模块，被拖曳到编辑区后会自动命名为 variable1，如图 6.19 所示。

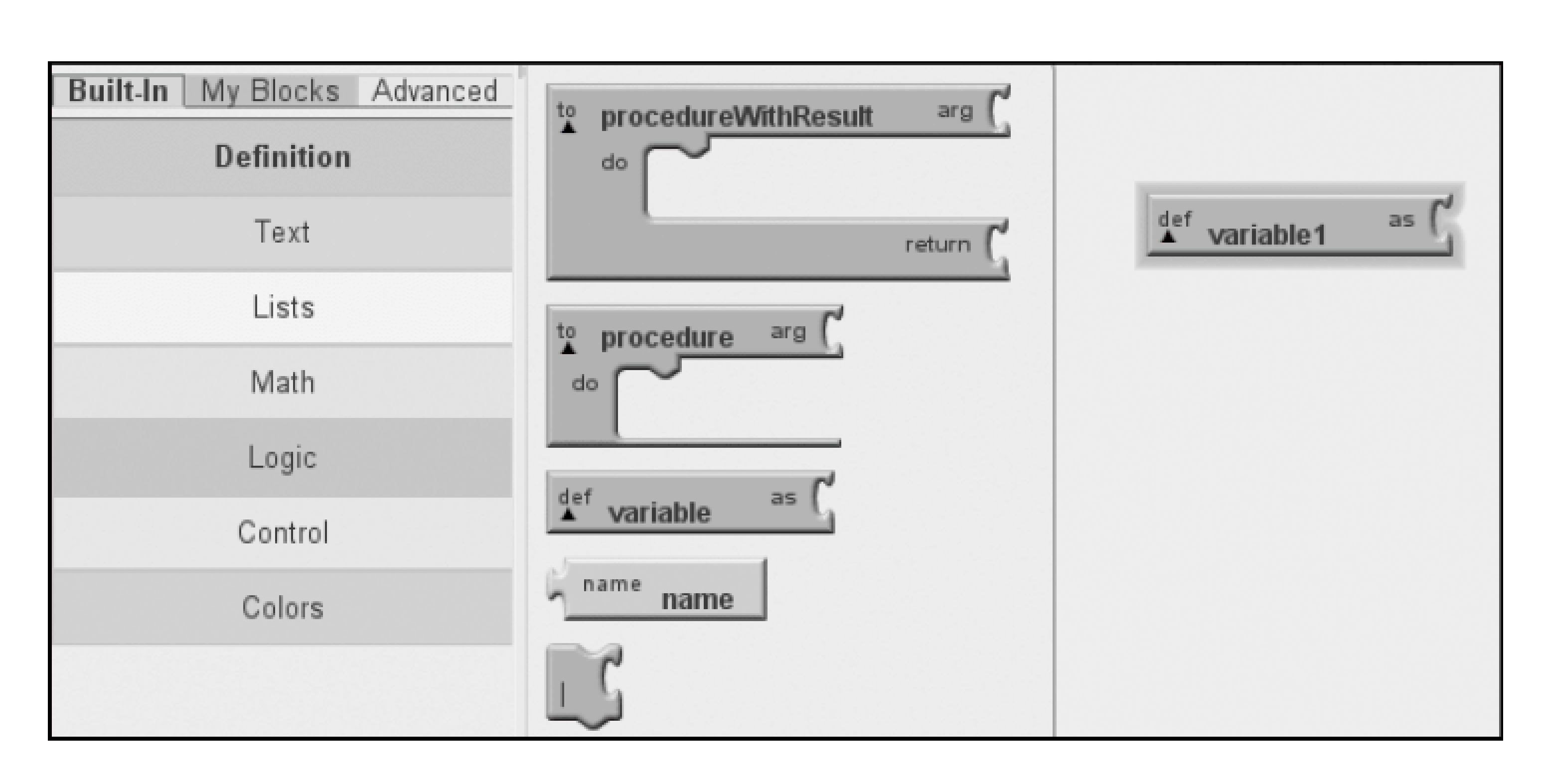


图 6.19　获取全局变量

单击全局变量的名称部分 variable1，将其更改为“paintsize”。通过 Built-In→Math→Number 添加一个数值模块，将数值更改为 4。将 paintsize 变量和数值模块组合，便将全局变量 paintsize 赋值为 4，如图 6.20 所示。paintsize 变量的值为 4，表示绘制圆形图案的半径为 4 个像素。

逻辑功能设计的第二步是响应“大圆形”按钮和“小圆形”按钮的单击事件，如图 6.21 所示。这两个按钮可以控制绘制圆形图案半径大小，“大圆形”按钮将半径设置为 10(即 paintsize=10)，“小圆形”按钮将半径设置为 4(即 paintsize=4)。

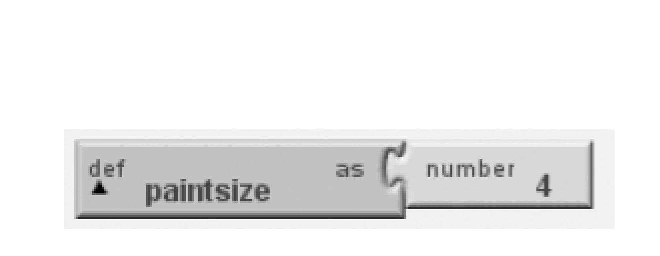


图 6.20　给 paintsize 全局变量赋值

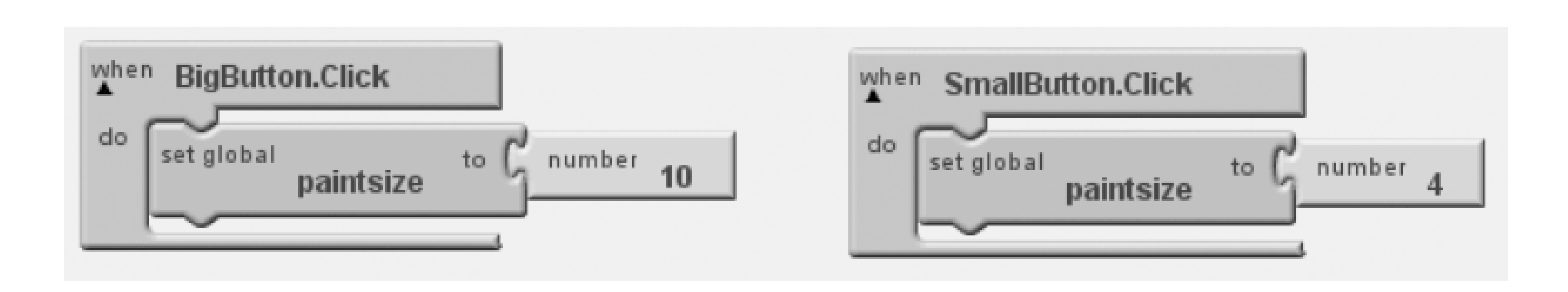


图 6.21　“大圆形”按钮和“小圆形”按钮的单击事件

逻辑功能设计的第三步是响应画布的触碰事件，也就是在画布上以 paintsize 变量作为半径，以触碰点(x,y)作为中心点绘制圆形图案，如图 6.22 所示。

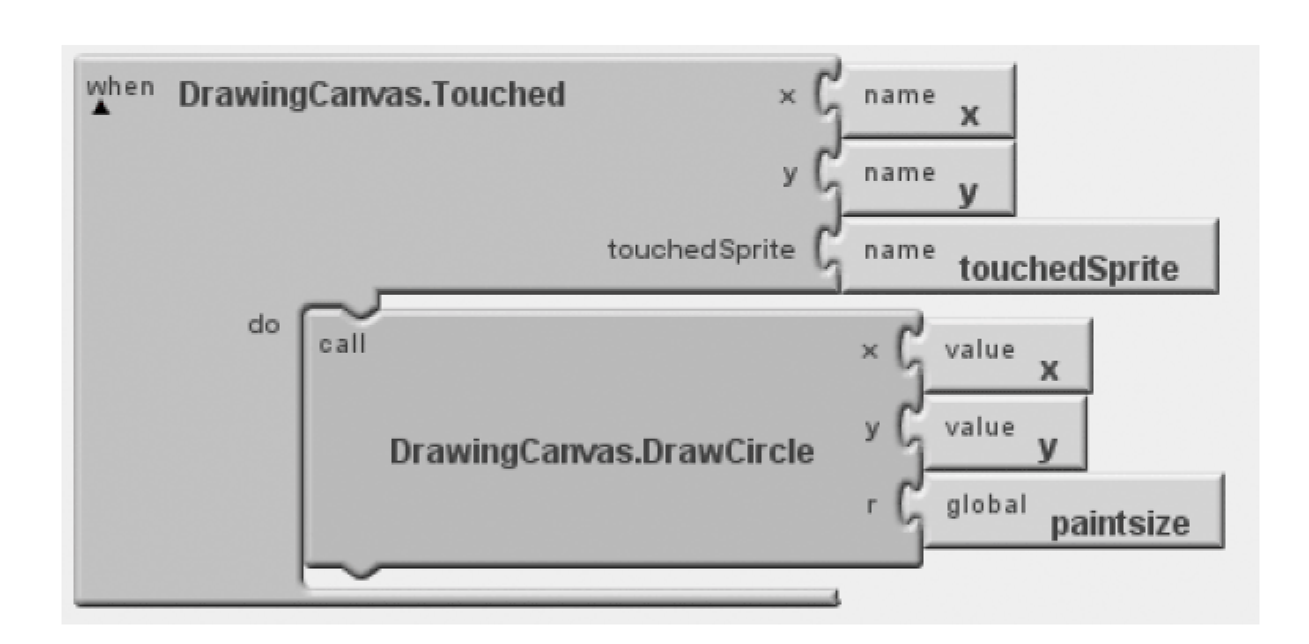


图 6.22　画布的触碰事件

在 My Blocks→DrawingCanvas→DrawingCanvas. Touched 获取到画布的触碰事件，与 My Blocks→DrawingCanvas→DrawingCanvas. DrawCircle 组合。再将 My Blocks→My

Definitions→ x、My Blocks → My Definitions → y 和 paintsize 全局变量组合到 DrawingCanvas. DrawCircle 模块上，这样就完成了画布触碰事件的处理。

逻辑功能设计的第四步是响应画布的拖曳事件，也就是在画布上按照手指移动的轨迹绘制线条，如图 6.23 所示。实现的方法是，在画布的拖曳事件中，根据当前点和前一个点的坐标绘制直线，因为拖曳事件的响应频率很高，这样在画面上看就是沿着手指移动形成的轨迹，而不是由多个点形成的折线。

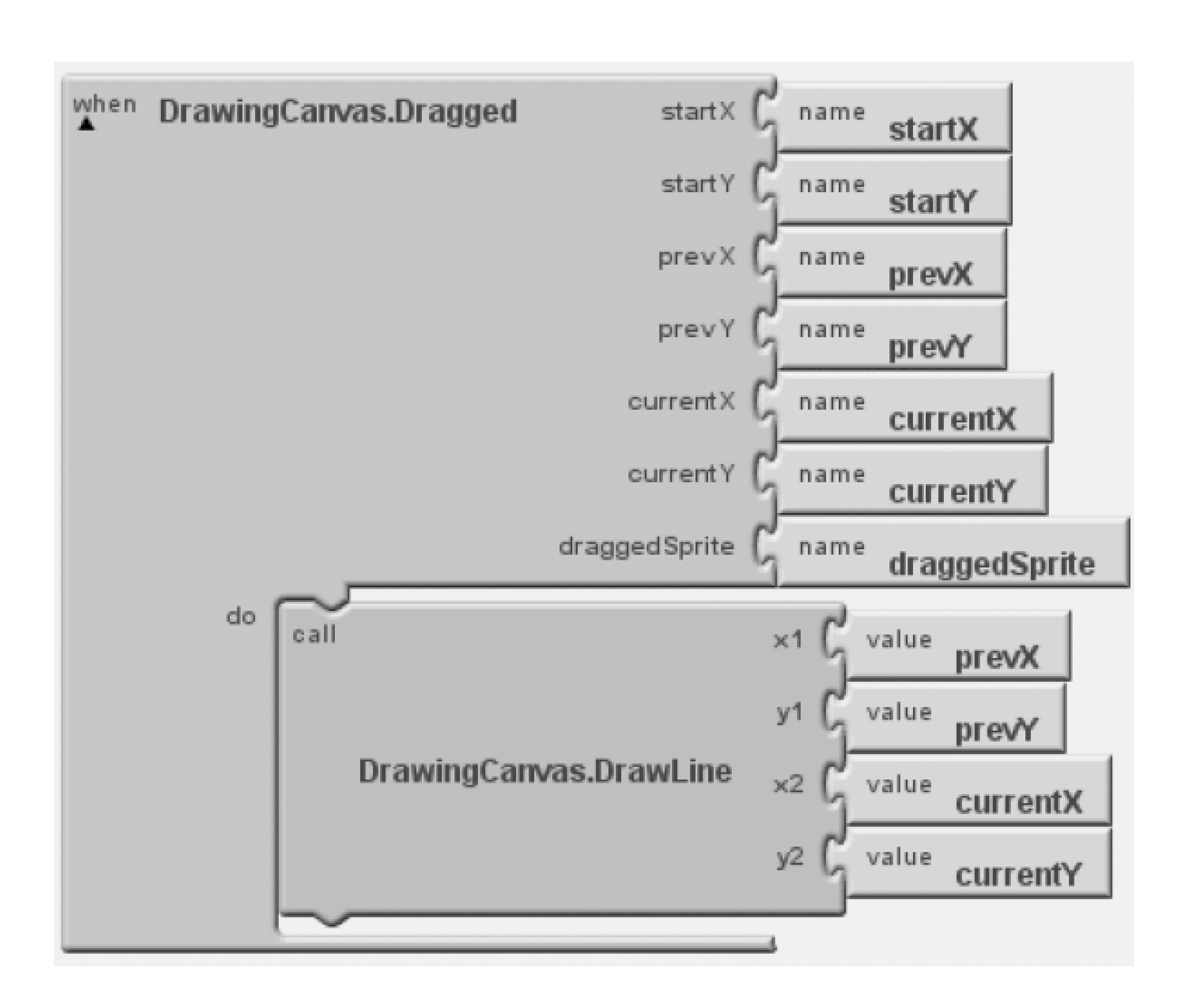


图 6.23 画布的拖曳事件

在 My Blocks→DrawingCanvas→DrawingCanvas. Dragged 获取到画布的拖曳事件，与 My Blocks→DrawingCanvas→DrawingCanvas. DrawLine 组合。再将 My Blocks→My Definitions → prevX、My Blocks → My Definitions → prevY、My Blocks → My Definitions → currentX 和 My Blocks → My Definitions → currentY 组合到 DrawingCanvas. DrawLine 模块上，这样就完成了画布拖曳事件的处理。

逻辑功能设计的第五步是响应三个修改画笔颜色按钮的事件，而且要在初始化的时候设置画笔的颜色。

先从 My Blocks→DrawingCanvas→DrawingCanvas. PaintColor 获取到控制画笔颜色的模块，在 Built-In→Colors 中分别获取表示红色、绿色和蓝色的模块，将颜色模块与 DrawingCanvas. PaintColor 属性组合在一起，就可以实现修改画笔的颜色。然后再将这些组合模块分别与 RedButton. Click、GreenButton. Click 和 BlueButton. Click 组合完成响应画笔颜色修改按钮的事件，如图 6.24 所示。

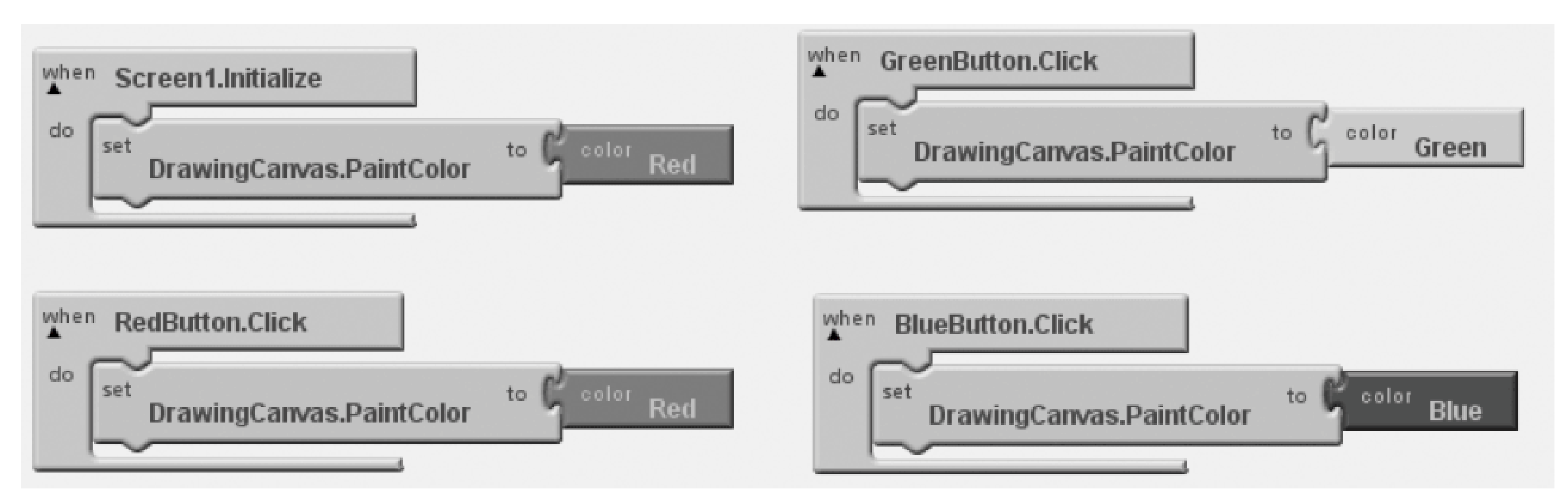


图 6.24 画笔颜色修改按钮的事件

画笔的默认颜色为黑色，而 PaintPic 示例中画笔只有红色、绿色和蓝色可选，这样需要在屏幕页初始化的时候修改画笔颜色，将画笔颜色修改为红色。

屏幕页初始化模块在 My Blocks→Screen1→Screen1. Initialize 中，该模块会在屏幕页启动时被调用，一般用来初始化控件的属性等操作。

逻辑功能设计的第六步是两种清空画布方法的实现，一种是单击“清空画布”按钮实现的，另一种是通过晃动手机实现的，如图 6.25 所示。

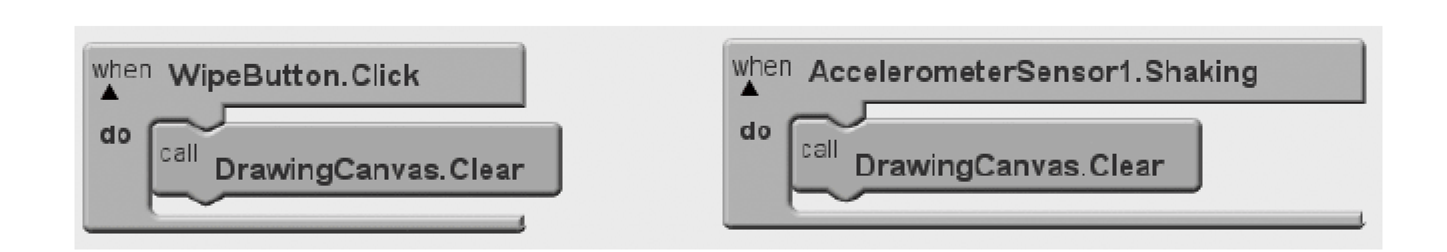


图 6.25　画布清空逻辑

清空画布的方法在 My Blocks→DrawingCanvas→DrawingCanvas. Clear，该模块会将画布上所有画笔所绘制的内容全部清除，但画布背景内容不会有变化。

加速度传感器的晃动事件在 My Blocks → AccelerometerSensor1 → AccelerometerSensor1. Shaking 中，该事件在晃动手机时被调用，经 DrawingCanvas. Clear 模块与其组合，就可以在晃动手机时清空画布内容。

逻辑功能设计的第七步是画布背景更换功能，同样是支持两种更换方法，一种是单击“选中图片”按钮，在手机的图片库里面选择画布背景；另一种是通过手机拍照，将拍摄的图片更换为画布背景，如图 6.26 所示。

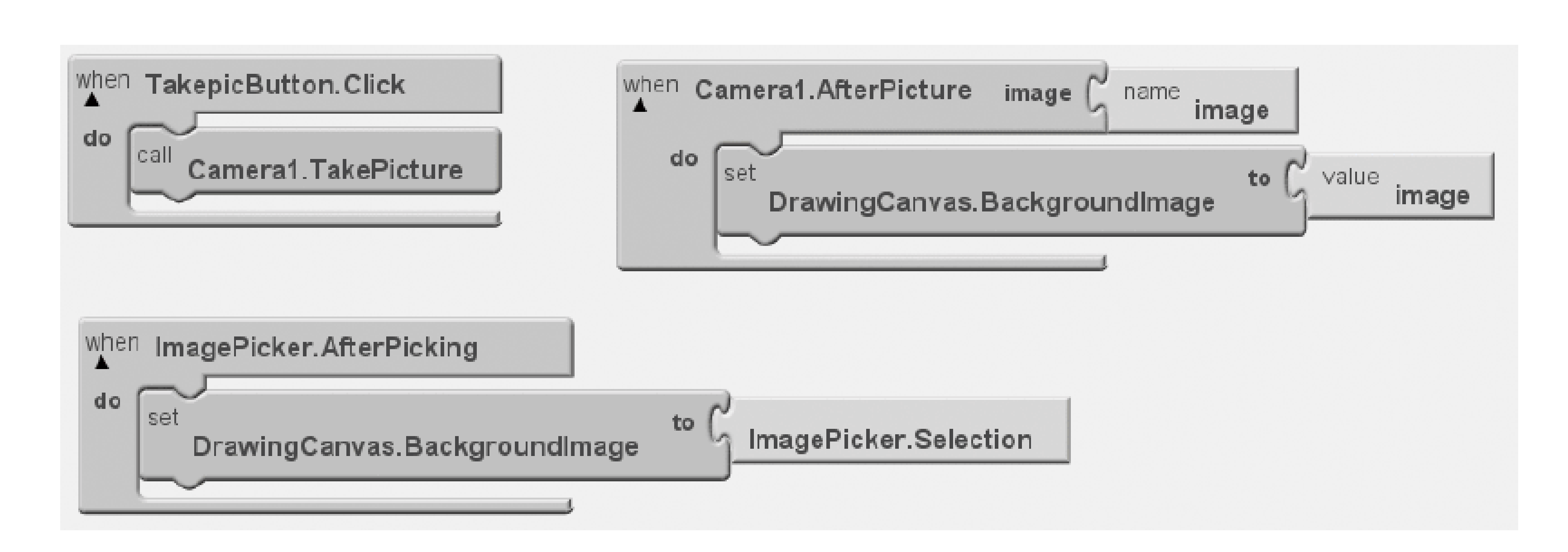


图 6.26　画布背景更换逻辑

更换画布背景图片是通过修改 My Blocks→DrawingCanvas→DrawingCanvas. BackgroundImage 属性实现的，将指定的图像(image)赋值给 BackgroundImage 属性，就完成了画布背景的修改。

除了从相机获取图像以外，还可以从 ImagePicker 获取背景图片。ImagePicker 的 AfterPicking 方法在用户选择图片后被调用，该方法与修改画布背景的模块组合在一起，并将 ImangePicker. Selection 作为图像参数传递给 BackgroundImage 属性，就可以实现画布背景的更改。

相机控件在用户单击 TakepicButton 按钮时，调用 TakePicture 方法拍照，在获取到照片后，会立即调用 AfterPicture 方法，用获取的照片修改画布背景。

逻辑功能设计的第八步，也是最后一步，就是将画布内容保存为文件。该方法的实现是通过调用 DrawingCanvas. Save 方法，直接将画布内容保存到手机的 SD 卡中，并将

该方法返回的文件路径传递给 SaveStatusLabel. Text 属性，显示在界面的最下方，如图 6.27 所示。

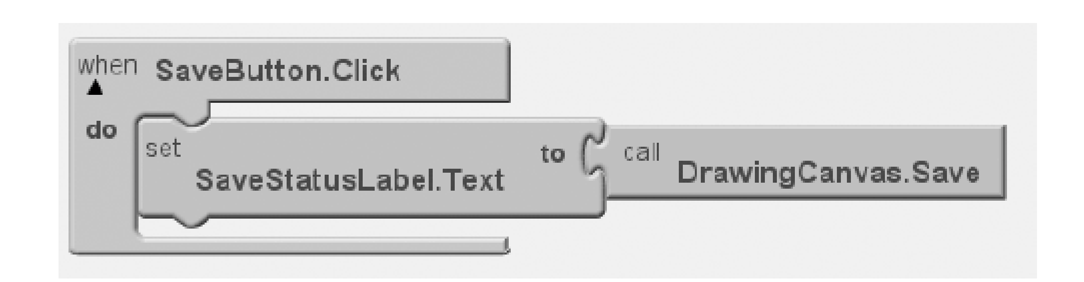


图 6.27 画布内容保存为文件

将上述逻辑模块添加完成后，在模块编辑器中全部逻辑功能如图 6.28 所示。

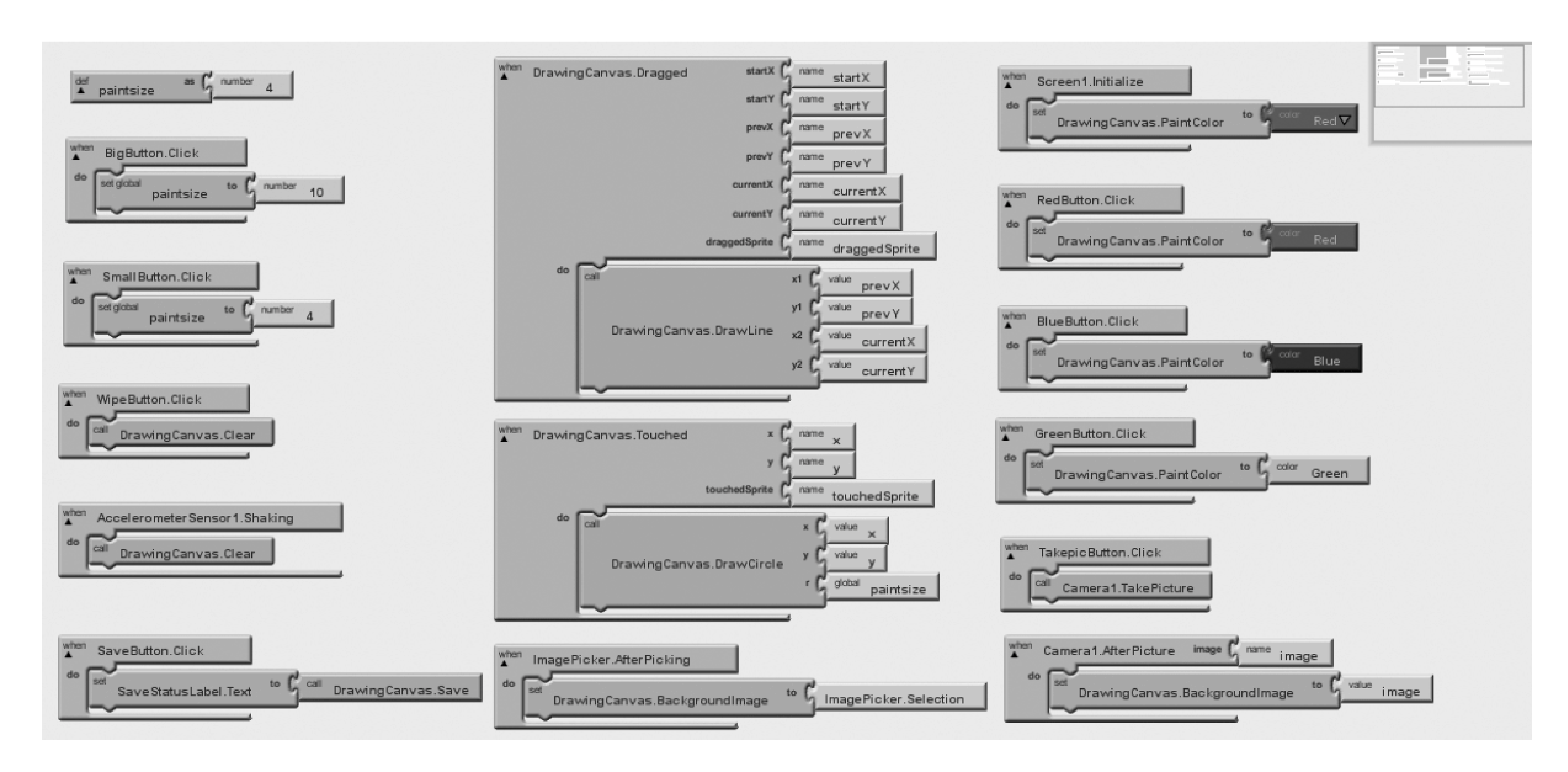


图 6.28 PaintPic 全部逻辑模块

PaintPic 示例在手机上的运行结果如图 6.29 所示。在模拟器上运行该示例时，则不可以使用手机拍照和通过晃动手机清空画布功能。

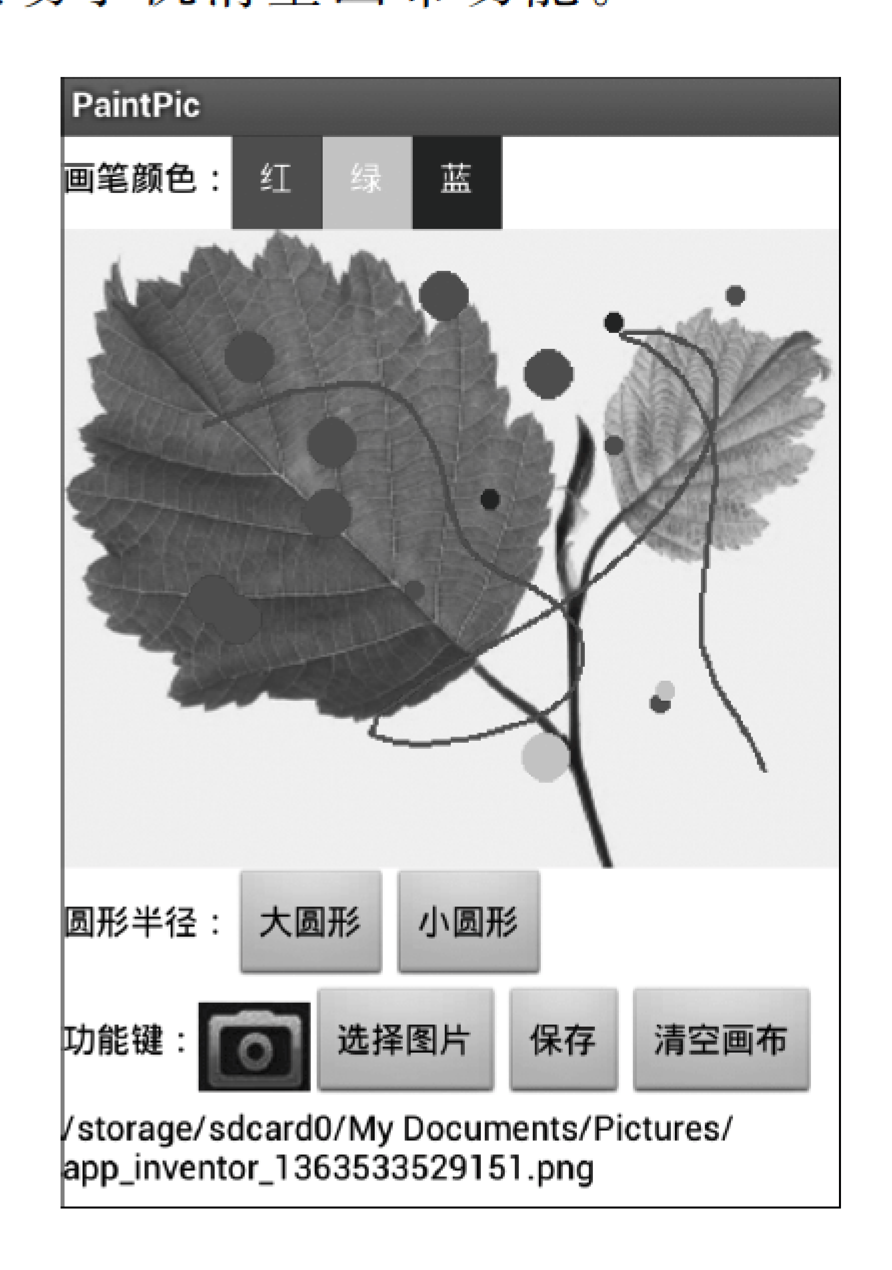
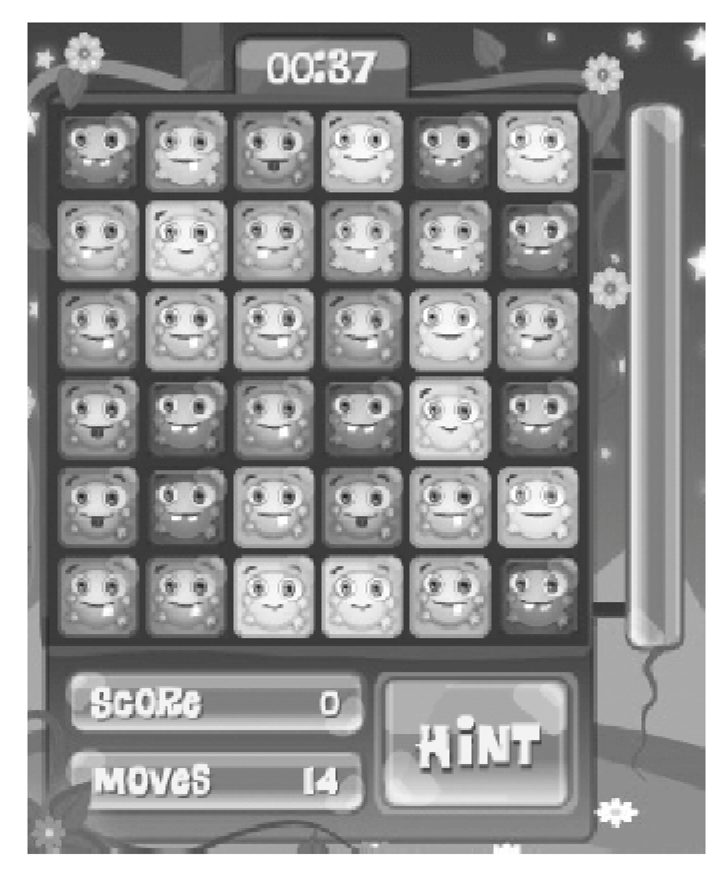


图 6.29 PaintPic 示例在手机上的运行效果

6.2 图像精灵

6.2.1 精灵介绍

图像精灵(ImageSprite)是一种可在画布中自由移动的图像,并可与球体(Ball)、其他图像精灵和画布边缘产生碰撞效果,因此图像精灵经常用于游戏开发,如图6.30所示。动画类别中的图像精灵如图6.31所示。



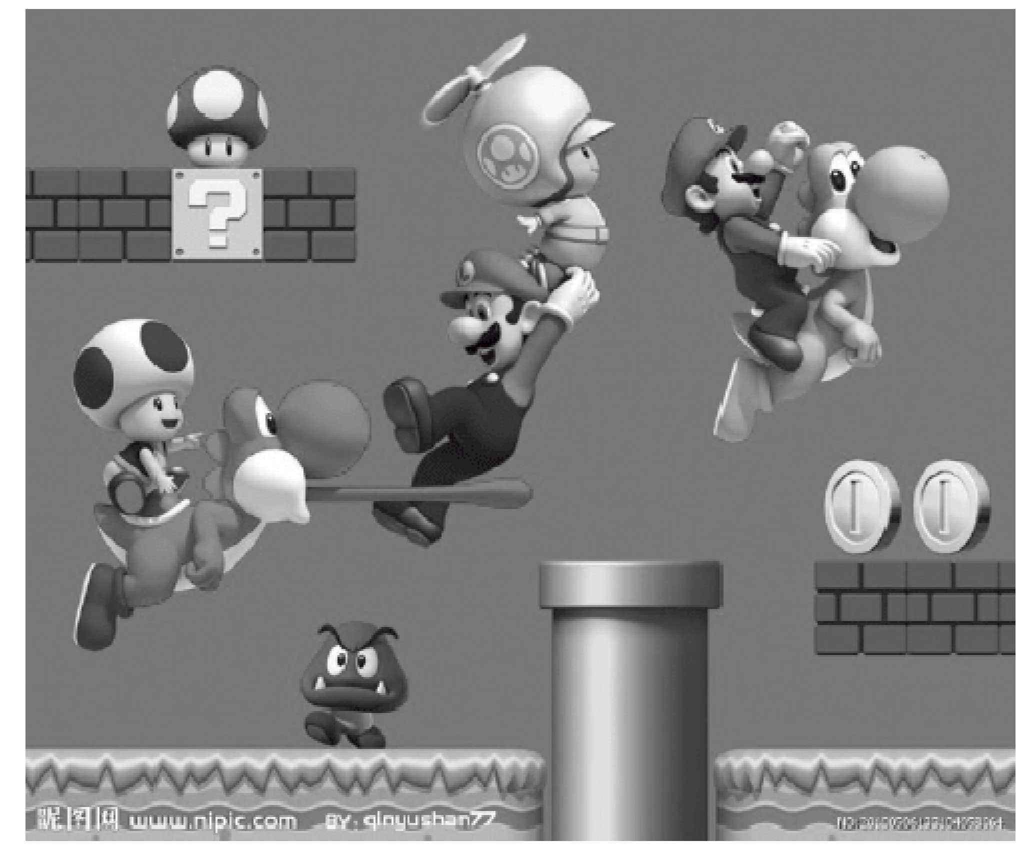


图6.30　游戏中的图像精灵

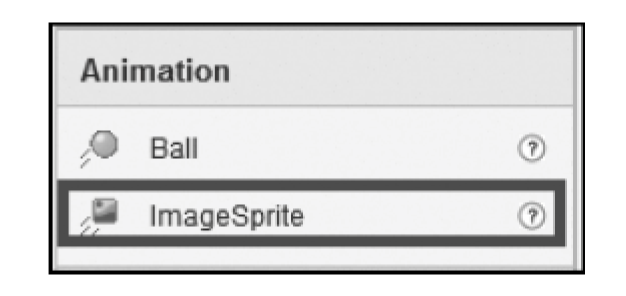


图6.31　动画类别中的图像精灵

6.2.2 精灵使用

将精灵放置在画布上,精灵就接受触摸或拖动操作,如果设置精灵自身关于运动的一些属性,精灵则可以按照预定的方式移动。

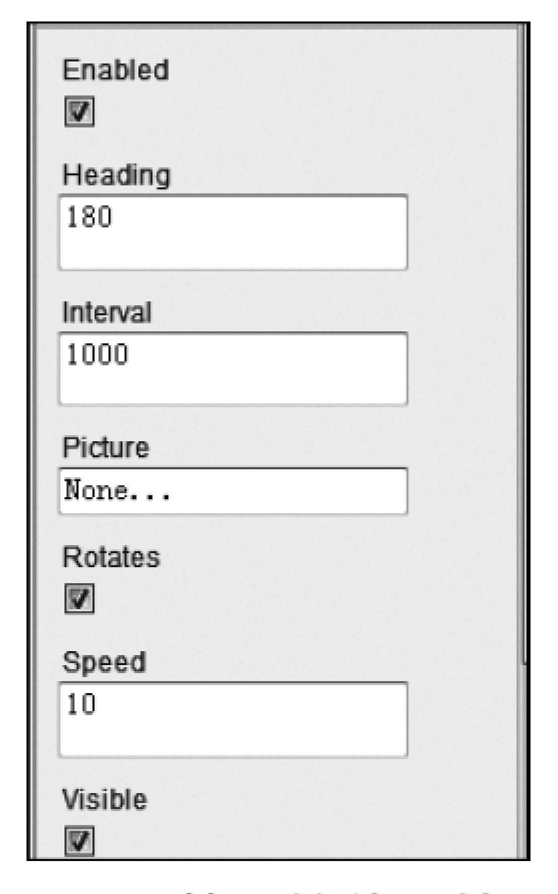


图6.32　精灵控件属性设置

例如,为实现一个图片精灵每隔1000ms向左侧移动10个像素,则需设置其速度(Speed)属性值为10像素,时间间隔(Interval)属性为1000ms,方向(Heading)属性设置为180°,激活(Enabled)属性设置为True。精灵的旋转属性(Rotates)设置为True时,图片会根据精灵的朝向变化进行转动,设置好的参数如图6.32所示。

图像精灵支持的属性如表6.5所示。其中,属性Picture需要指定一个图片,是精灵在画布上的图像。属性Rotates表示是否允许精灵旋转,如果允许,图像精灵在改变移动方向(Heading)时,图片会自动旋转以匹配新的移动方向。

X和Y是图像精灵在画布上的坐标,X为0时,表示图像精灵已经到达画布的左侧边界;Y为0时,则表示图片精灵已经到达画布的上边界。Enabled属性表示图像精灵是否会被激活,如果精灵与移动相关的属性被设置,同时Enabled为true,则图像精灵会在

画布上移动。Visible 属性决定了图像精灵在画布上是否可见。

表 6.5 精灵的属性

属 性	说 明	属 性	说 明
Picture	精灵图片	X	所在位置横坐标
Enabled	是否激活	Y	所在位置纵坐标
Interval	移动频率	Width	图片宽度
Rotates	是否允许精灵旋转	Height	图片高度
Heading	移动方向	Visible	是否可见
Speed	移动速度		

属性 Heading 是图像精灵的移动方向，取值范围是 0～360，其中，0 表示水平向右，90 代表垂直向上，180 代表水平向左，270 表示垂直向下，如图 6.33 所示。

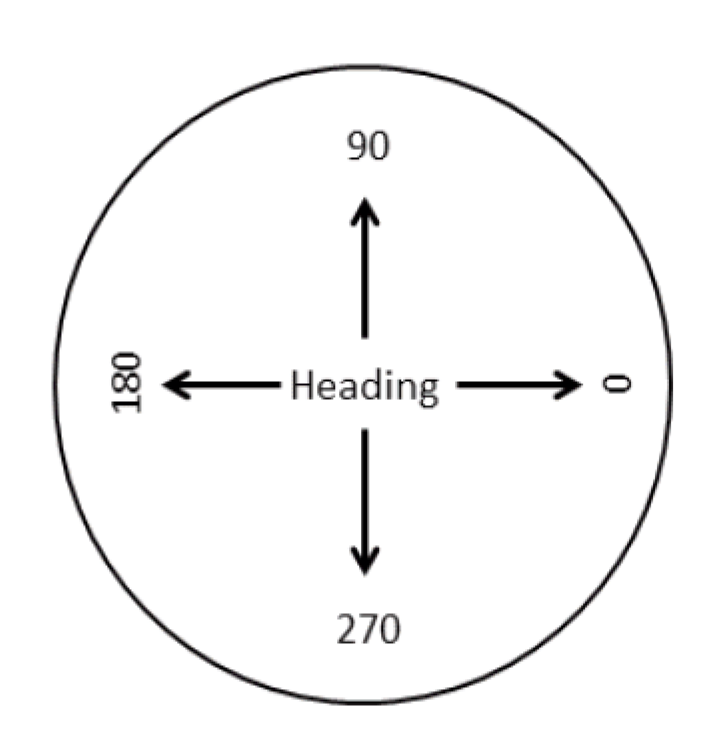


图 6.33 精灵朝向数值代表含义

图片精灵支持的事件包括碰撞事件、拖曳事件、触壁事件、非碰撞事件和触摸事件等，所支持的全部事件如表 6.6 所示。

表 6.6 精灵的事件

事 件	说 明	事 件	说 明
CollidedWith	碰撞事件	Dragged	拖曳事件
EdgeReached	触壁事件	Flung	快速划动事件
NoLongerCollidingWith	不再碰撞事件	TouchUp	触碰抬起事件
Touched	触摸事件	TouchDown	触碰按下事件

图片精灵的事件模块如图 6.34 所示。碰撞事件、不再碰撞事件和触壁事件属于碰撞检测事件，这部分内容将在“高级动画功能”小节介绍。

精灵控件的方法实现了精灵的移动、反弹和碰撞检测动作。图像精灵支持的方法如表 6.7 和图 6.35 所示。

ColldingWith 方法用来检测图像精灵是否发生碰撞，返回值为 true 时，说明与指定精灵或画布边缘发生碰撞；返回值为 false 时，则未发生碰撞。

MoveIntoBounds 方法是将超出边缘的控件，重新移动至画布边缘范围内。

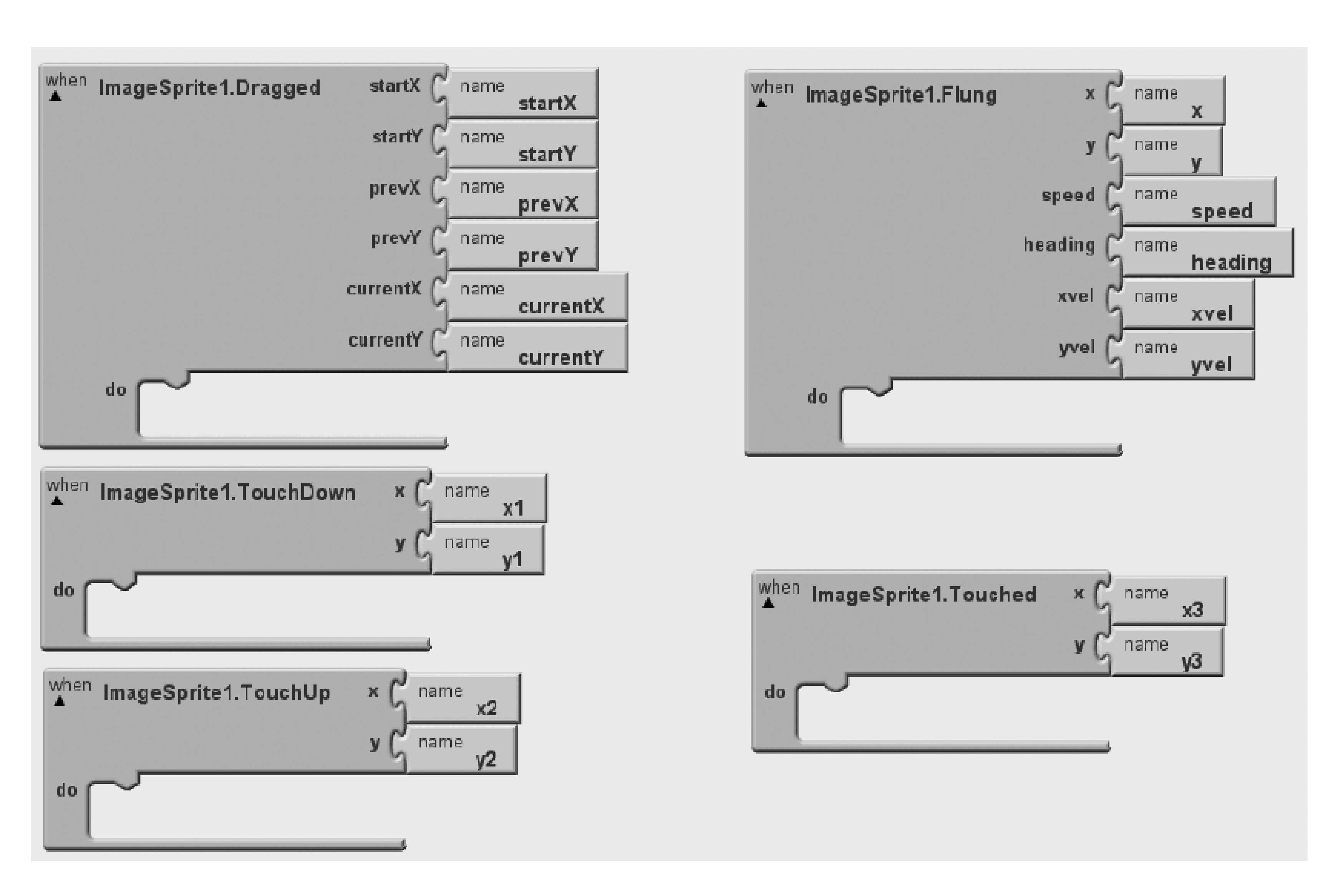


图 6.34　精灵控件事件

表 6.7　精灵的方法

方　法	说　　明
Bounce	精灵从边缘反弹
ColldingWith	检测精灵是否碰撞
MoveIntoBounds	精灵超出边界时将精灵移至边界内
MoveTo	将精灵移动到指定坐标点
PointTowards	将移动方向朝向另一个精灵
PointInDirection	将移动方向朝向指定坐标点

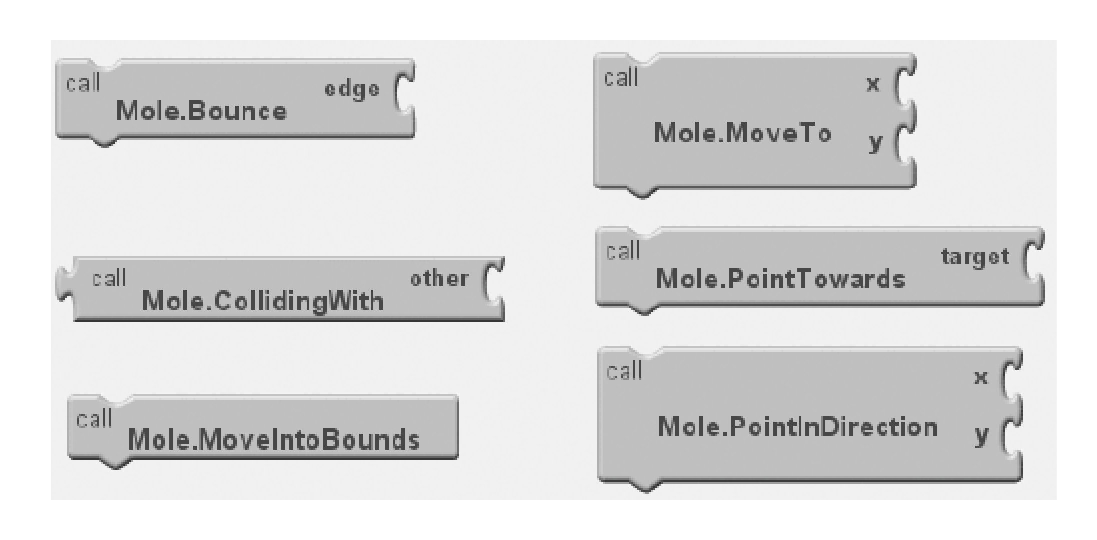


图 6.35　精灵的方法

6.2.3　示例——打地鼠

Mole 示例是经典的“打地鼠”游戏，鼹鼠在 5 个洞中随机出现，但出现的时间非常短暂，因此要集中精力快速单击洞口的鼹鼠，每次成功地单击鼹鼠得 1 分，积累到 10 分游戏胜利。Mole 示例的运行界面如图 6.36 所示。

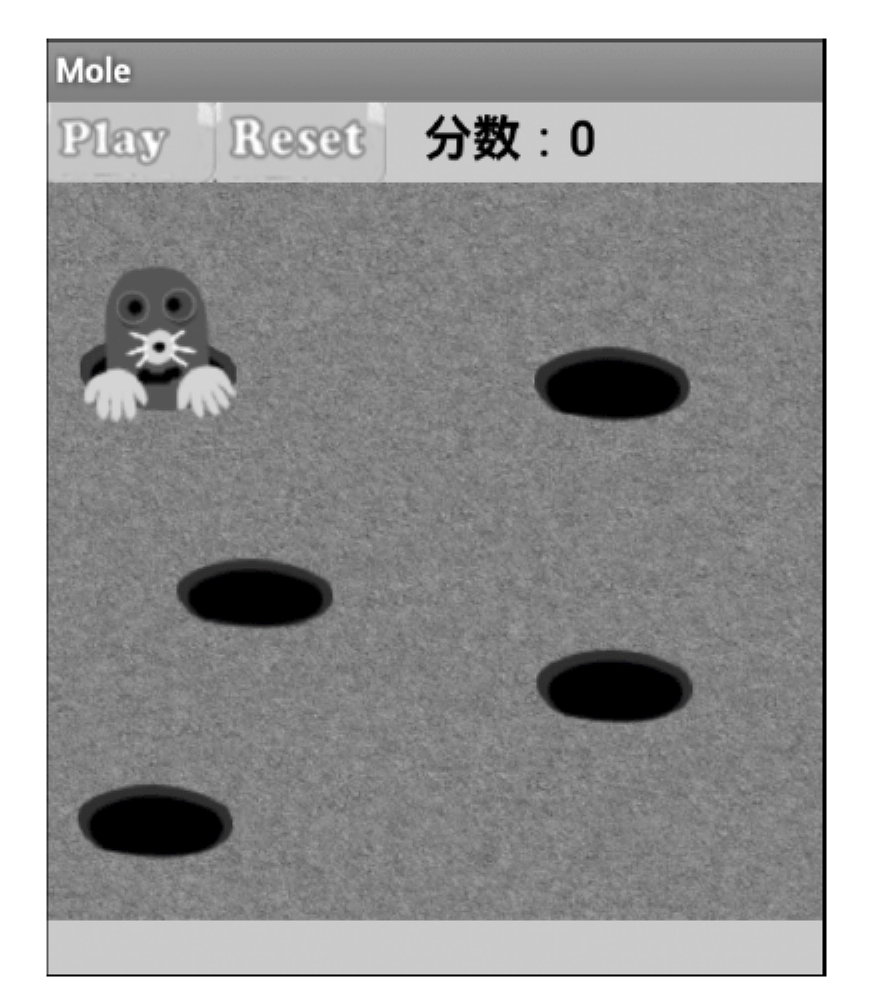


图 6.36　Mole 示例的运行界面

单击 Play 按钮开始游戏，鼹鼠会随机出现在洞口，每次单击鼹鼠将会产生一次振动反馈，同时“分数”加 1。在游戏进行中，如果用户单击 Reset 按钮，分数将清零，再重新单击 Play 按钮后重新开始游戏。游戏积累到 10 分后，将出现“恭喜你，你赢了！”的游戏胜利提示，并发出“叮咚”声音，如图 6.37 所示。

在 Mole 示例的资源中，共有 5 张图片和 1 个声音文件，这些资源都是游戏中不可缺少的，资源的说明如表 6.8 所示。

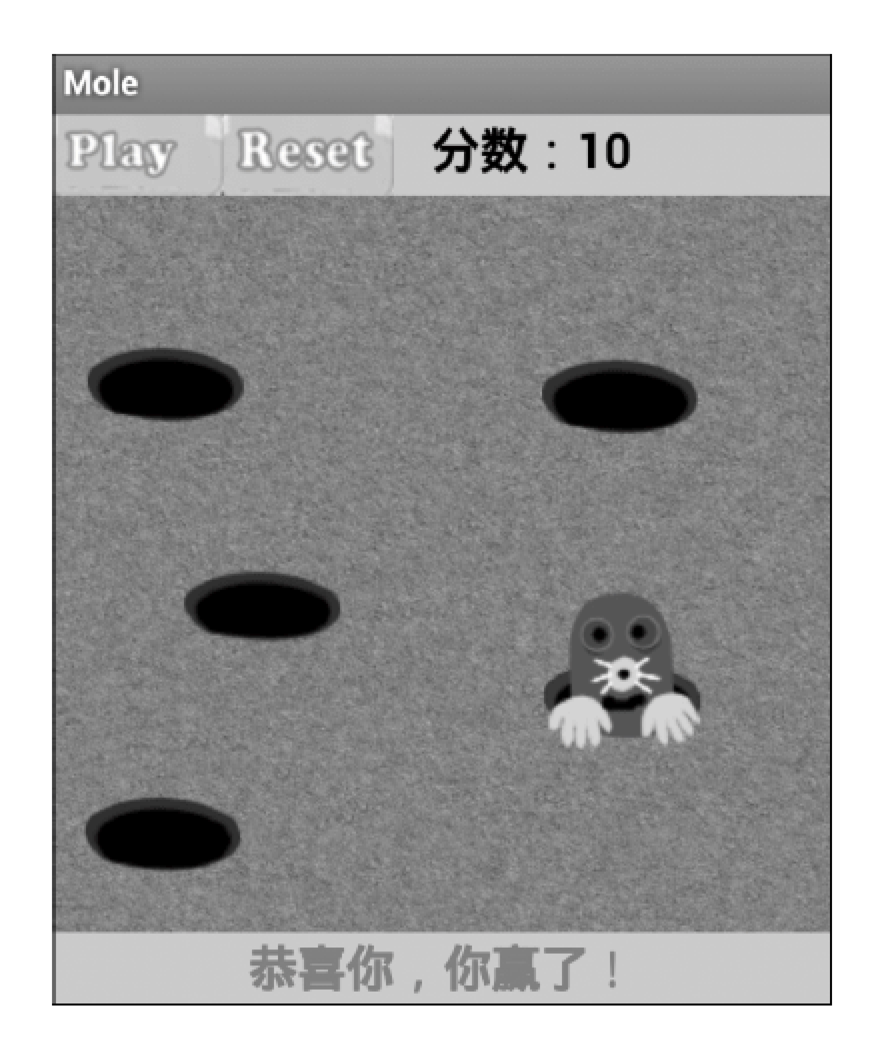


图 6.37 游戏获胜画面

表 6.8 资源文件说明

文 件	说 明
PlayButton.png	Play 按钮图片
ResetButton.png	Reset 按钮图片
grass.jpg	Canvas1 画布的背景图片
hole.png	洞口精灵的图片
mole.png	鼹鼠精灵的图片
DingDong.mp3	游戏获胜的“叮咚”声音

Mole 示例的界面设计图如图 6.38 所示。

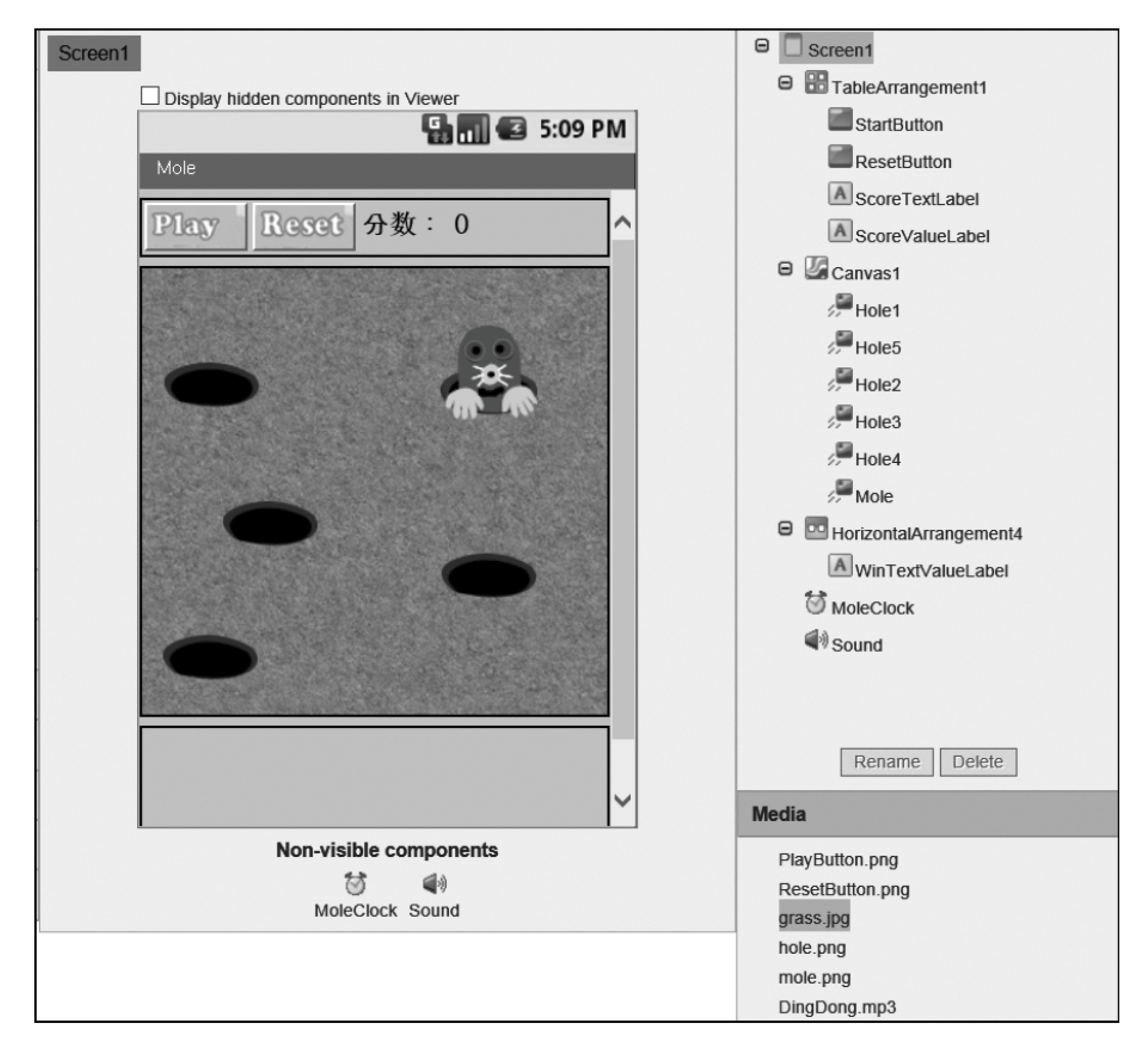


图 6.38 Mole 示例的界面设计图

时钟 MoleClock 控制着鼹鼠精灵周期性地移动，因此设置 MoleClock 时钟的 Enable 属性为 false，让时钟在用户单击 Play 按钮后再启动。设置时间间隔（TimerInterval）为 10 000ms（10s），TimeAlwaysFires 设置为 true。

将声音控件 Sound 的 Source 属性设置为已经上传的 DingDong.mp3 文件，读者也可以选择自己喜欢的音频，但需要控制上传文件的大小。将标签 WinTextValueLabel 的

Enable 属性设为 false，在用户游戏胜利时再显示该标签内容。

在模块编辑器中，首先定义两个全局变量 holes 和 currentHole，如图 6.39 所示。holes 是表示洞口图像精灵的列表，在定义时先调用 make a list 方法获得一个空列表，在屏幕页初始化时动态添加元素。currentHole 表示当前洞口的图像精灵，在自定义的 MoveMole 函数中使用。

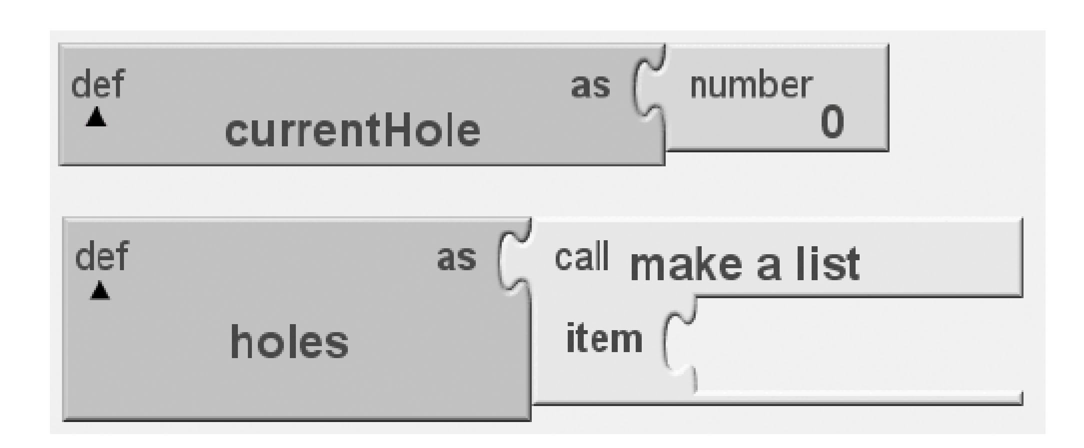


图 6.39　定义全局变量

屏幕页的初始化事件完成了三项工作，如图 6.40 所示：①初始化 5 个表示洞口的图像精灵的 Picture 属性；②设置时钟 MoleClock 的 TimeEnabled 属性为 false；③设置图片精灵 Mole 的 Enabled 属性为 false。

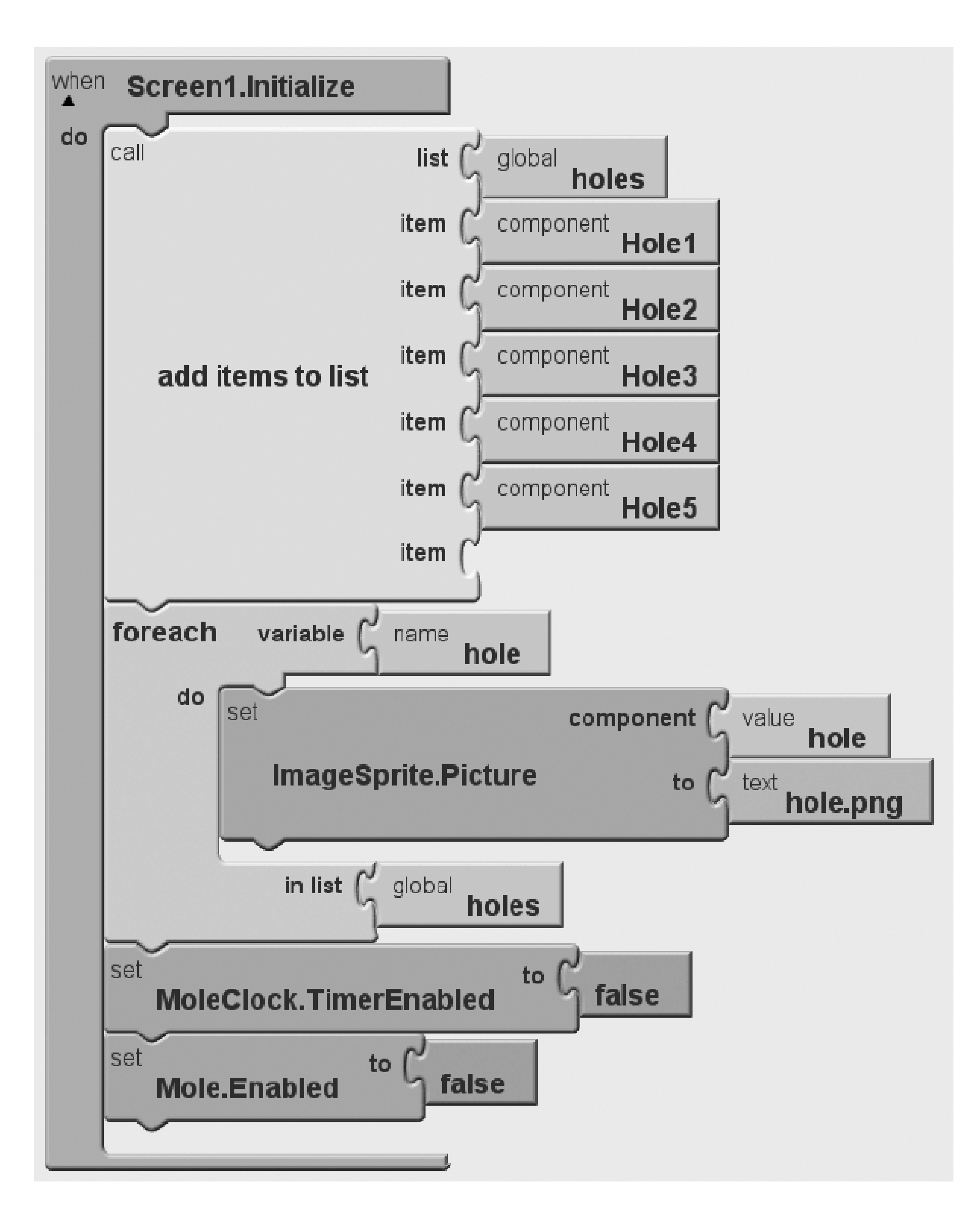


图 6.40　屏幕页 Screen1 的 Initialize 事件

虽然在界面编辑器中已经设置了时钟 MoleClock 的 TimeEnabled 属性为 false，为了避免误操作引起逻辑错误，仍然在屏幕页的初始化事件再次对该属性进行设置。

设置图像精灵 Mole 的 Enabled 属性为 false，可以让图像精灵暂时不接受触碰（Touched）事件，避免游戏还没有正式开始前，用户已经可以开始通过单击鼹鼠获得游戏分数。

初始化图片精灵 Hole1～Hole5 有两种方法：①在界面编辑器中设置其 Picture 属性为上传文件 hole.png；②在屏幕页初始化函数中动态修改 Hole1～Hole5 的 Picture 属性。本示例使用的是第二种方法，此方法略显繁琐，但却演示了一种动态的、批量的修改图像精灵属性的方法。

这里用到了 Advanced→Any ImangeSprite→ImageSprite. Picture 模块，如图 6.41 所示。Advanced 区的模块可以对所有同类型的控件进行操作，比如修改所有标签的文字、修改所有按钮的颜色等。

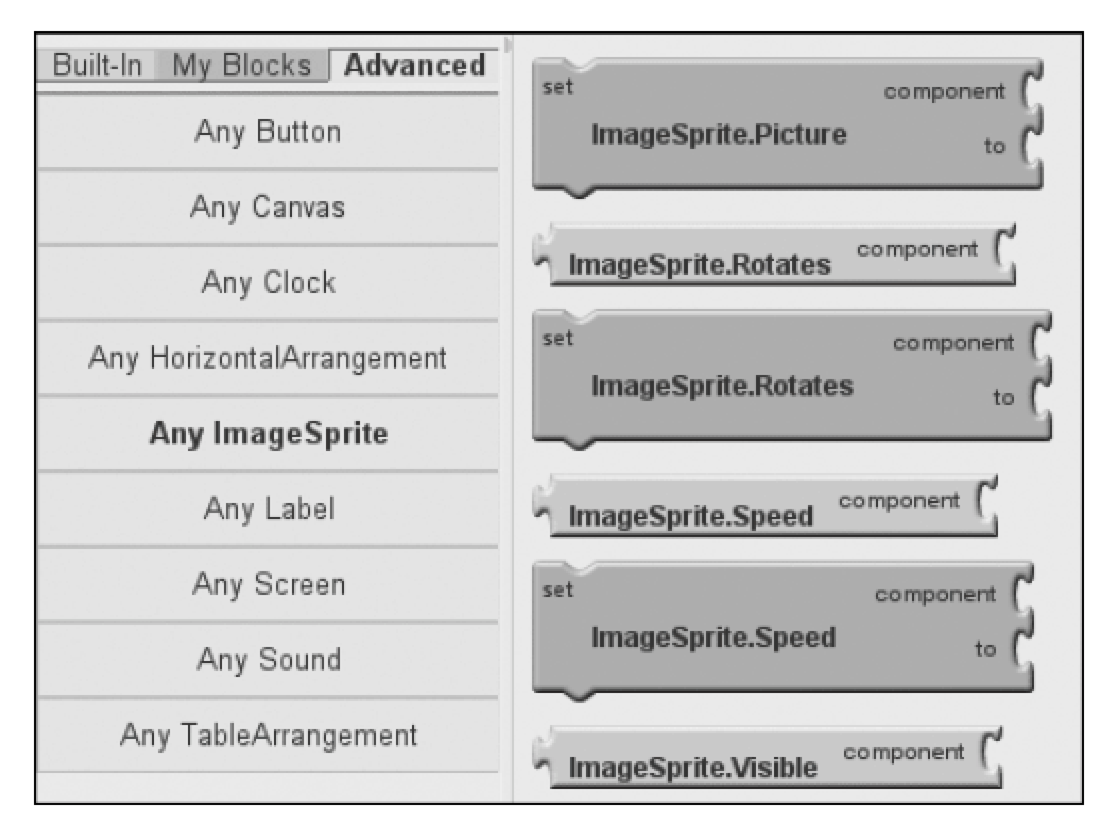


图 6.41　Advanced 区的模块

ImageSprite. Picture 模块，如图 6.42 所示，槽 component 用来拼接图像精灵(ImageSprite)模块，槽 to 用来拼接图片(Picture)属性。Advanced 区的其他模块有着与 ImageSprite. Picture 模块相似的结构和槽，可以分别拼接各自支持的控件类型和属性值。

全局变量 holes 的初始化是通过调用 Built-In→Lists→add items to list 方法，将所有表示洞口的图像精灵控件加载到全局变量 holes 中，如图 6.43 所示。

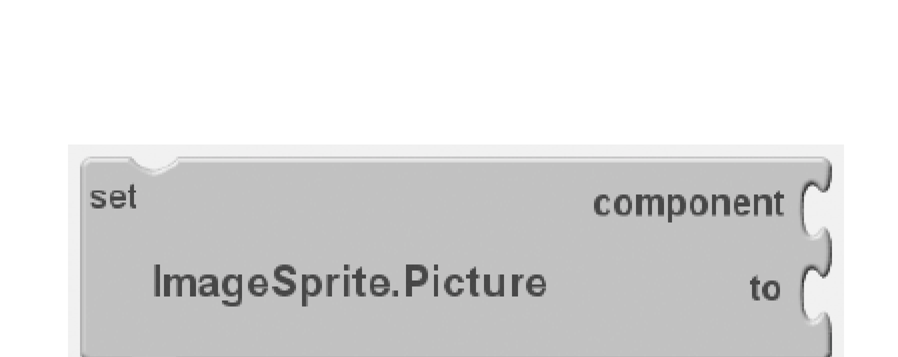


图 6.42　ImageSprite. Picture 模块

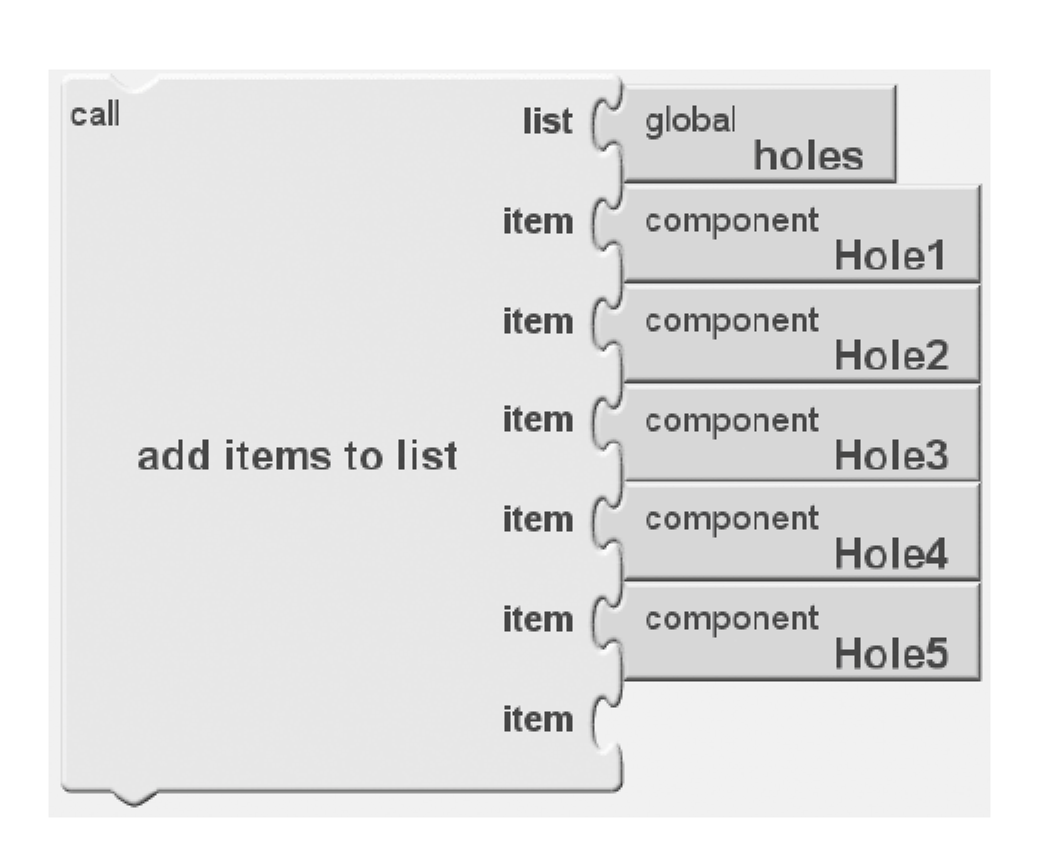


图 6.43　初始化 holes 列表

在 foreach 循环中，调用 ImageSprite. Picture 模块，将每个表示洞口图像精灵的 Picture 属性都赋值为 hone. png，如图 6.44 所示。

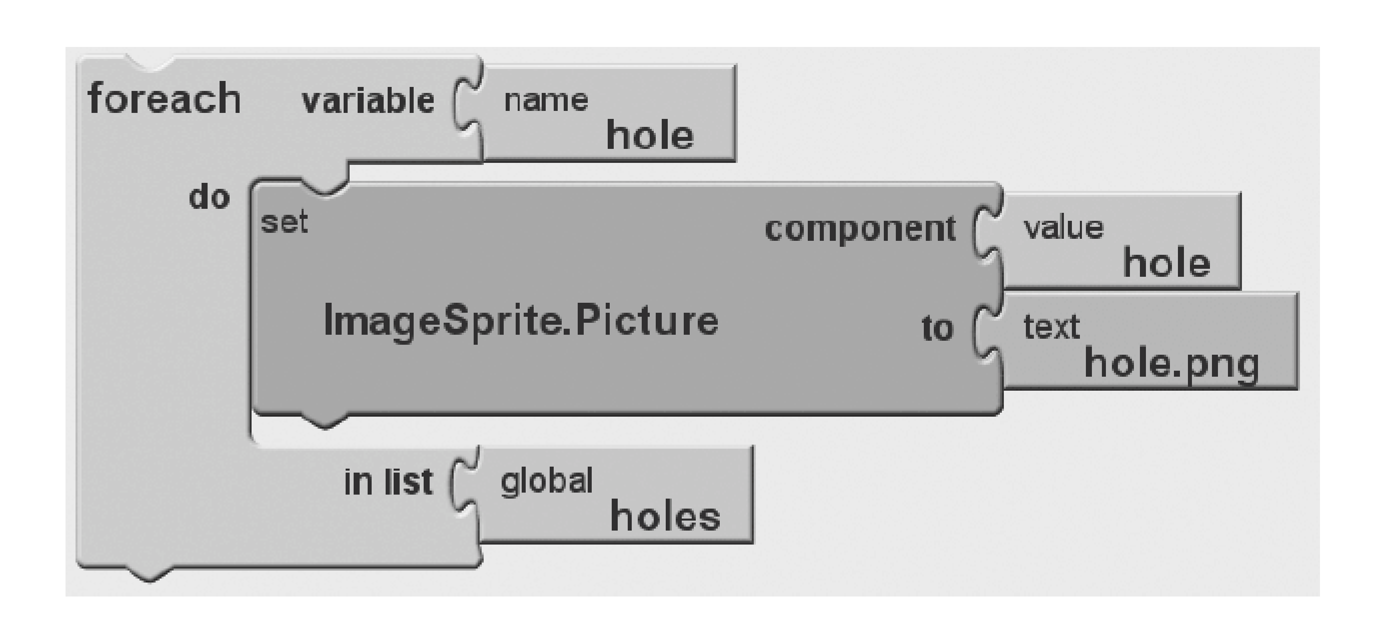


图 6.44　循环调用 ImageSprite. Picture 模块

为了重复利用已有代码，将鼹鼠的移动写成函数 MoveMole，这个函数将鼹鼠的图像精灵随机地出现在 5 个洞口中的任意一个，逻辑模块如图 6.45 所示。

在 MoveMole 函数中，首先调用 pick random item 方法，在 holes 列表中随机选取一个图像精灵，赋值给全局变量 currentHole。然后调用鼹鼠图像精灵 Mole 的 MoveTo 方法，将 currentHole 中图像精灵的坐标 X 和坐标 Y 作为参数，传递给 MoveTo 方法，这样

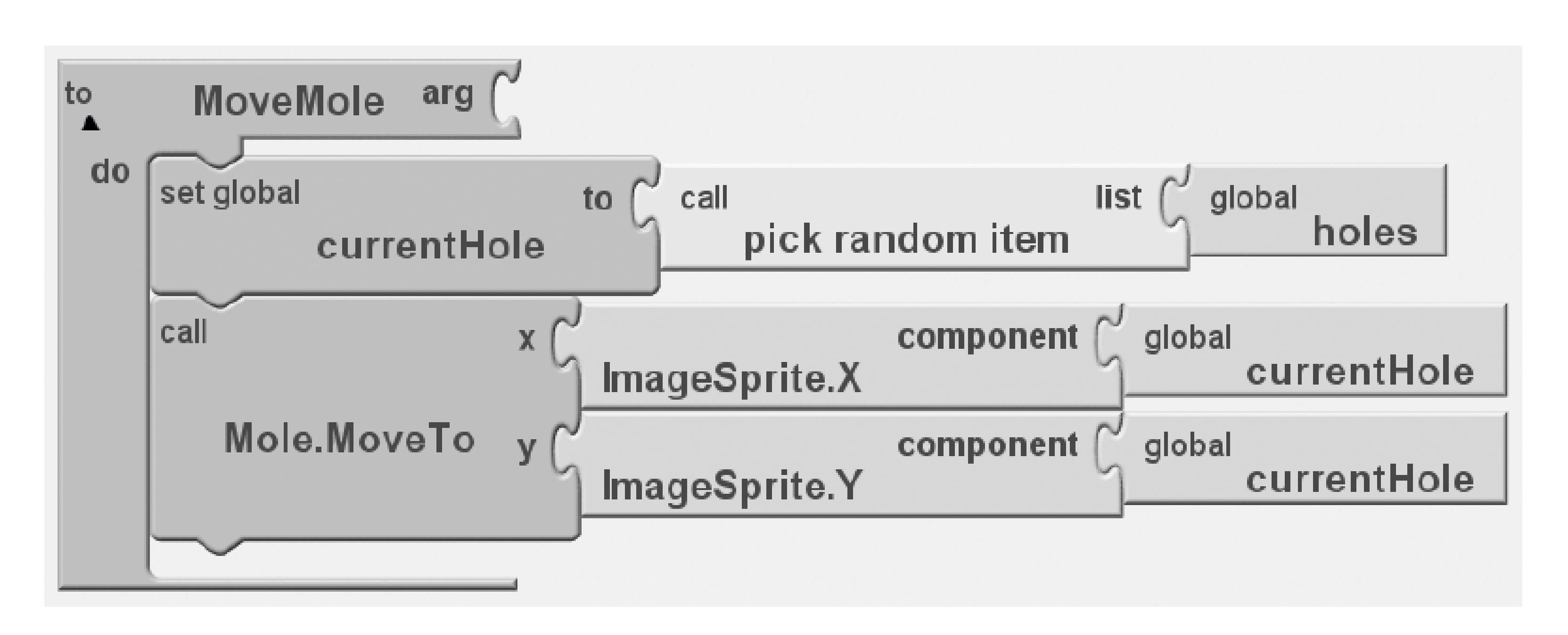


图 6.45 MoveMole 函数

就可以将鼹鼠图像精灵移动到全局变量 currentHole 所代表的洞上面。

在时钟 MoleClock 的触发事件中，只调用 MoveMole 函数，移动表示鼹鼠的图像精灵，如图 6.46 所示。

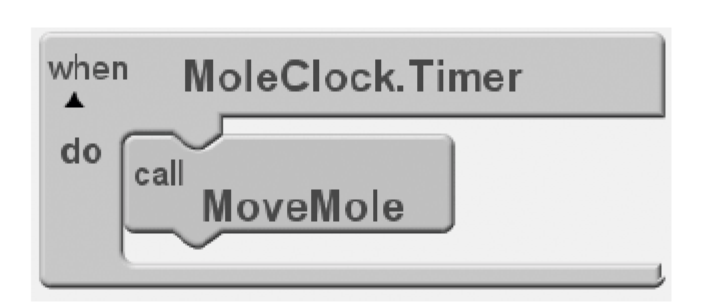


图 6.46 MoveClock 时钟的 Timer 方法

在游戏初始化阶段，MoleClock 时钟是不运行的，只有用户单击 Play 按钮后，MoleClock 时钟才开始运行。在单击 Reset 按钮，或者用户游戏胜利后，时钟 MoleClock 又变为不运行状态。时钟是否运行，由属性 TimeEnabled 控制。在图 6.47 中，可以找到控制时钟 MoleClock 是否有效的模块。

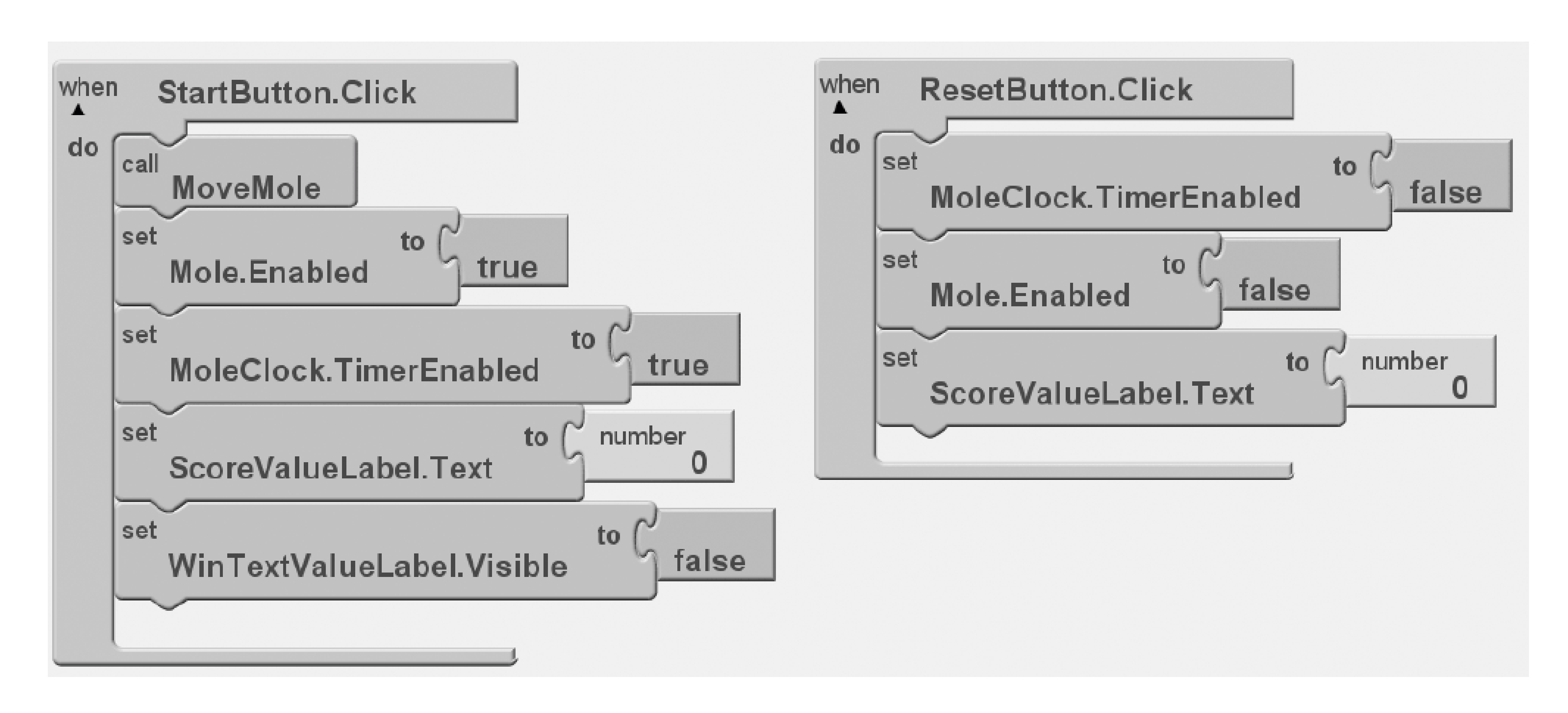


图 6.47 两个按钮单击事件

表示鼹鼠的图像精灵 Mole 的属性 Enabled 的逻辑与时钟 MoleClock 是一样的，也是在用户单击 Play 按钮后允许用户单击(true)。在单击 Reset 按钮后，变为不可单击(false)。

按钮事件中还完成了分数归零和隐藏胜利信息等功能，如图 6.47 所示。

因为游戏启动后，时钟 MoleClock 要在一个周期以后才会触发，在触发事件中鼹鼠才会移动。为了在游戏启动后立即产生鼹鼠移动的效果，要在 StartButton 的单击事件中主动调用 MoveMole 函数，完成鼹鼠的一次移动。

在图像精灵 Mole 的 Touched 事件(图 6.48)中,用户每次触碰精灵,都要调用一次 MoveMole 函数,移动一次鼹鼠,并发出振动反馈(Sound. Vibrate)。如果游戏分数(ScoreVaueLabel. Text)小于 10,则每次触碰都会将游戏分数增加 1。如果游戏分数已经大于等于 10,则游戏结束,发出游戏胜利的音效(Sound. Play),并停止时钟(MoleClock),使鼹鼠图像精灵失效(Enabled 为 false),并显示游戏胜利的消息(WinTextValueLable. Visible 为 true)。

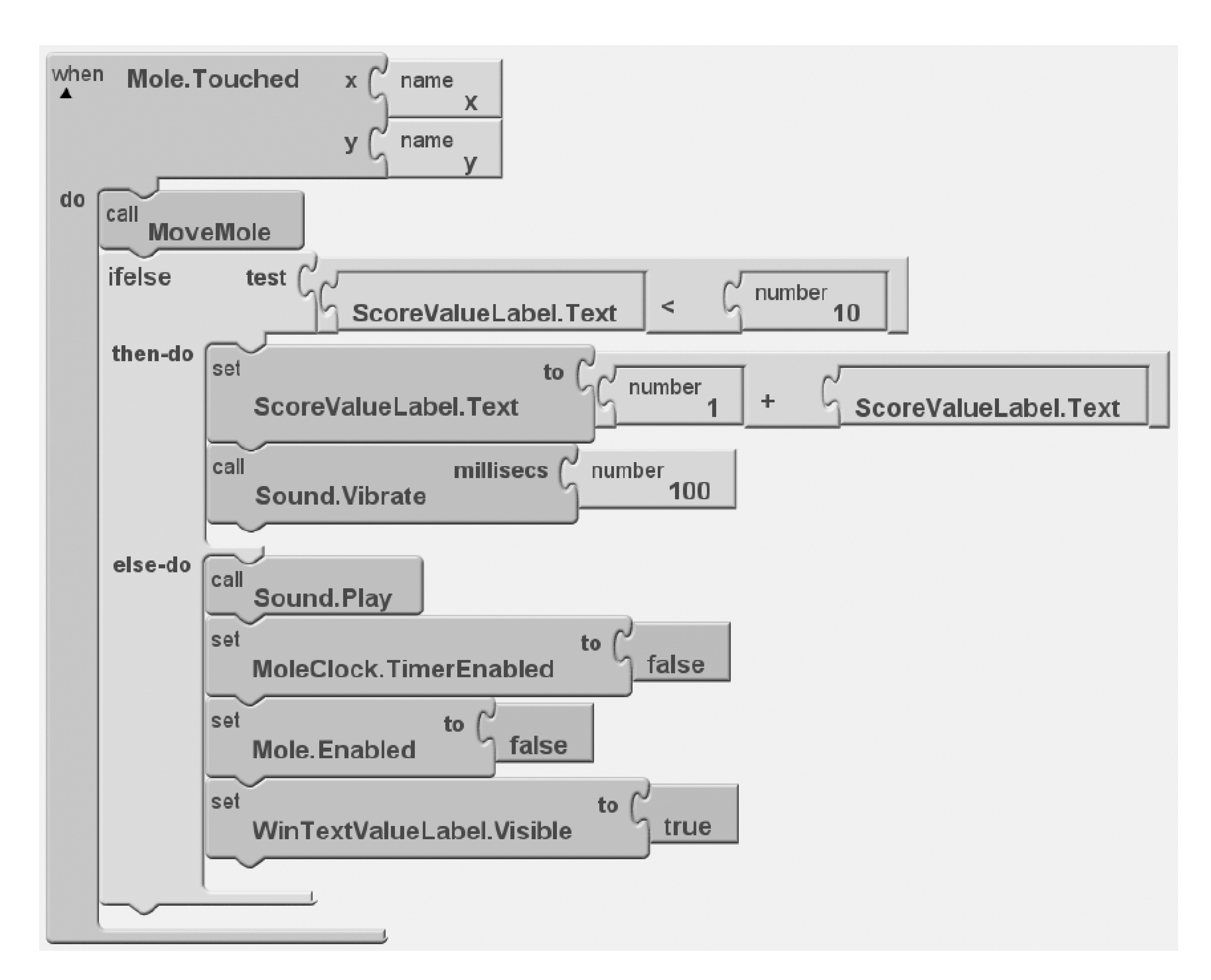
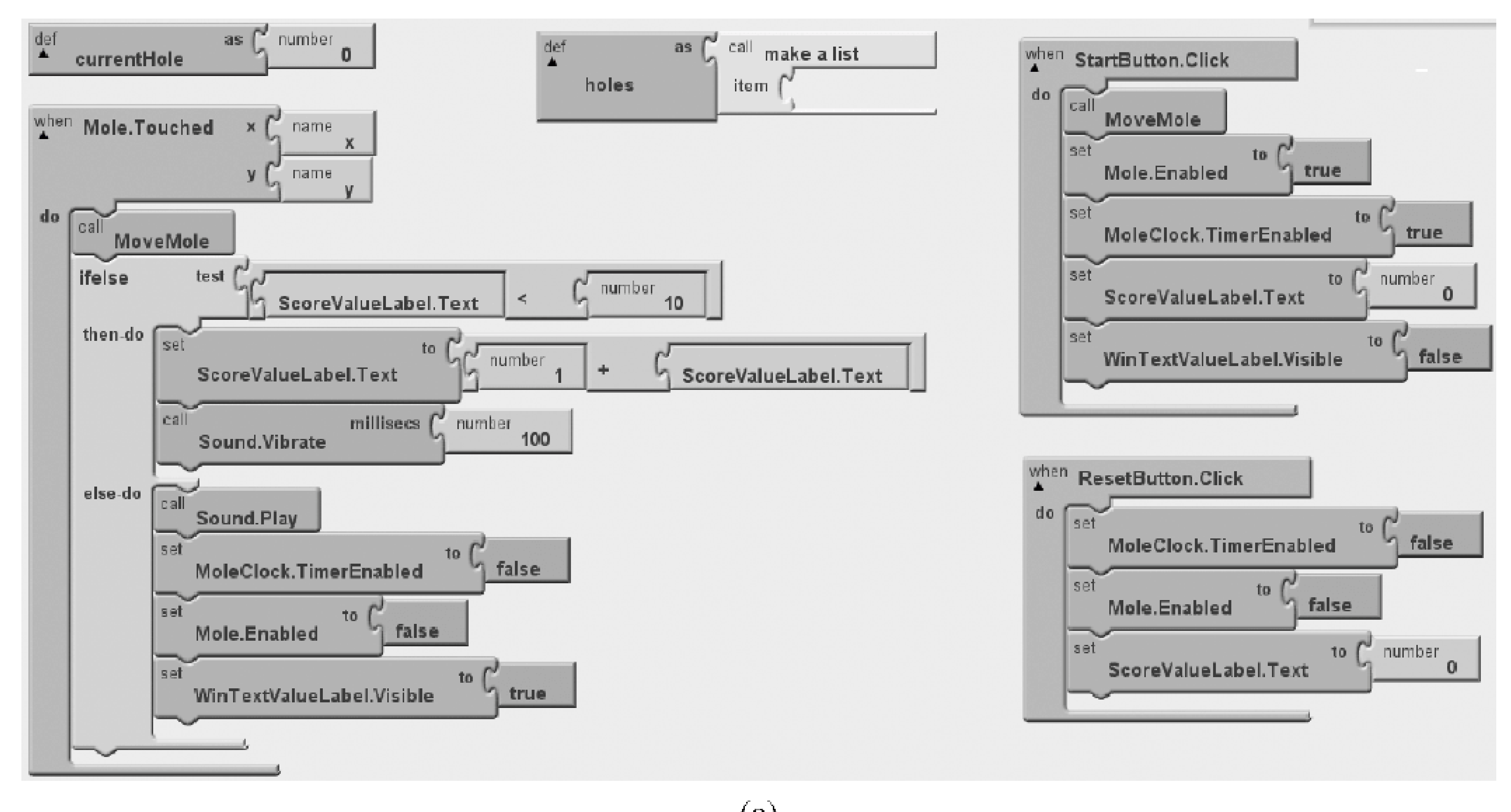


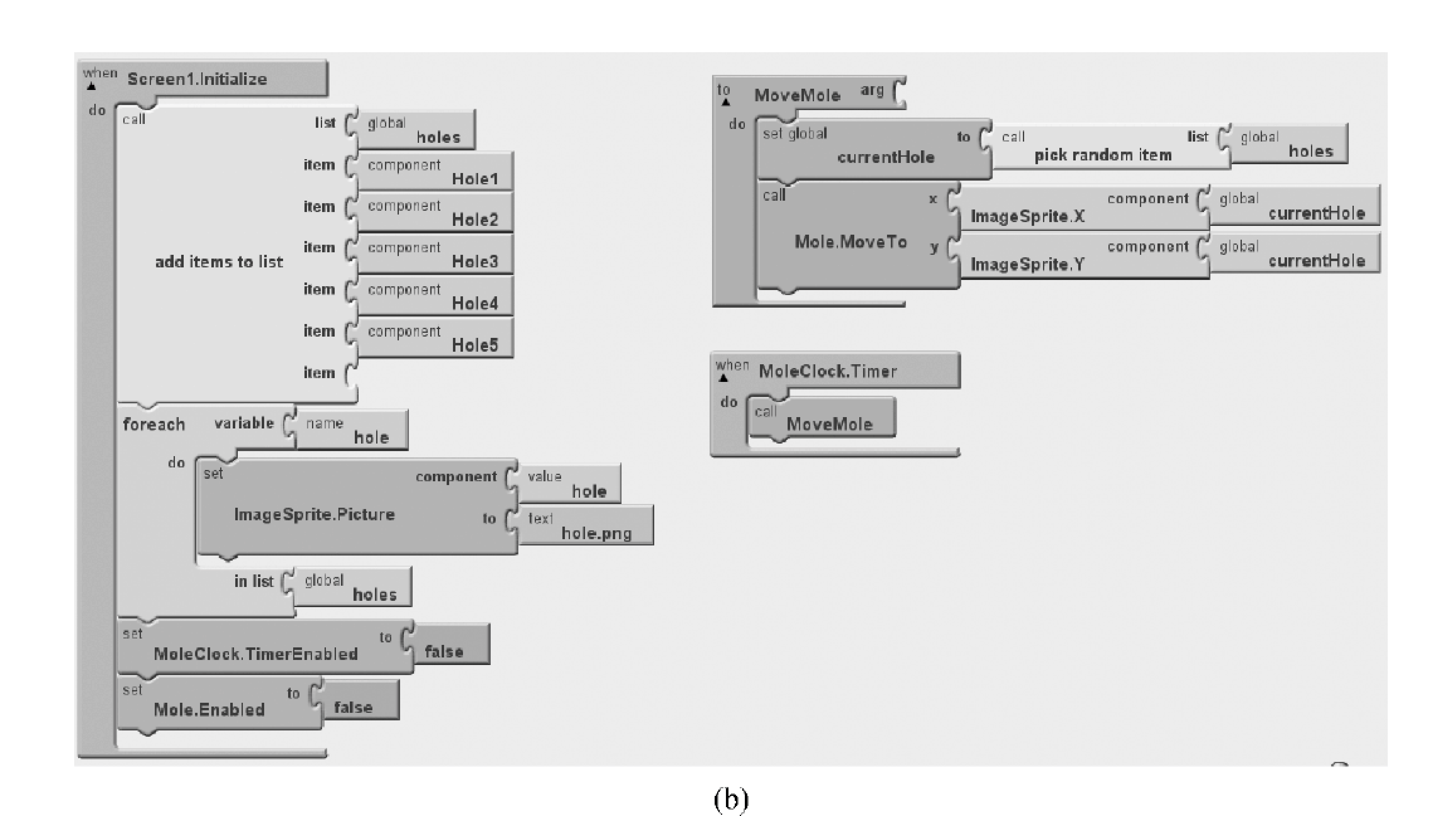
图 6.48 图像精灵 Mole 的 Touched 事件

Mole 示例的全部逻辑模块如图 6.49 所示。



(a)

图 6.49 Mole 示例的全部逻辑模块



(b)

图 6.49 （续）

6.3 高级动画功能

6.3.1 碰撞检测

碰撞检测是在精灵的运动过程中，检测到精灵自身边缘与其他精灵或画布边缘接触的技术，在开发游戏过程中经常会被使用到。

图 6.50 是疯狂小鸟的游戏截图，这其中就大量使用到了碰撞检测技术。例如，小鸟在飞行过程中碰撞到石块，小鸟和石块都要有相应的反应，小鸟会弹回去，石块会裂开。再例如，两个摞在一起的小猪，上方的小猪和下方的小猪只可以边界互相接触，而不能够重合。



图 6.50 疯狂小鸟游戏画面

碰撞检测技术的实现要运用数学和物理知识，在不同的情况下采用不同的碰撞检测

方式，本节主要介绍碰撞检测的基本原则和使用方式。

一般情况下，只有游戏中的物体发生移动后才有必要进行碰撞检测，所以碰撞检测的流程分为三步：更新物体位置→进行碰撞检测→碰撞处理。

实现碰撞检测通常包括三个方面的内容。首先，确定检测对象。游戏在运行中会有很多实体对象，在进行碰撞检测时并不需要对所有的实体对象都检测一遍，如静止的宝箱没有必要去检测和另外的宝箱是否发生了碰撞。所以在开始碰撞检测之前，首先要确定碰撞检测的对象是什么。其次，检测是否碰撞，这是检测的核心环节。在这个环节需要综合考虑游戏本身的需求，以及运行平台的性能等问题，合理地选择碰撞检测的算法。最后是处理碰撞，当检测到碰撞发生的时候，就需要根据碰撞的类型进行相应的处理，例如，炮弹会在碰到目标后爆炸并给目标带来伤害。

6.3.2 球的使用

球体(Ball)是特殊的图像精灵，具有与图像精灵相似的属性、完全相同的事件和方法。不同之处在于球体不能够有自定义的外观，但可以改变球的大小和颜色。球体控件如图 6.51 所示。

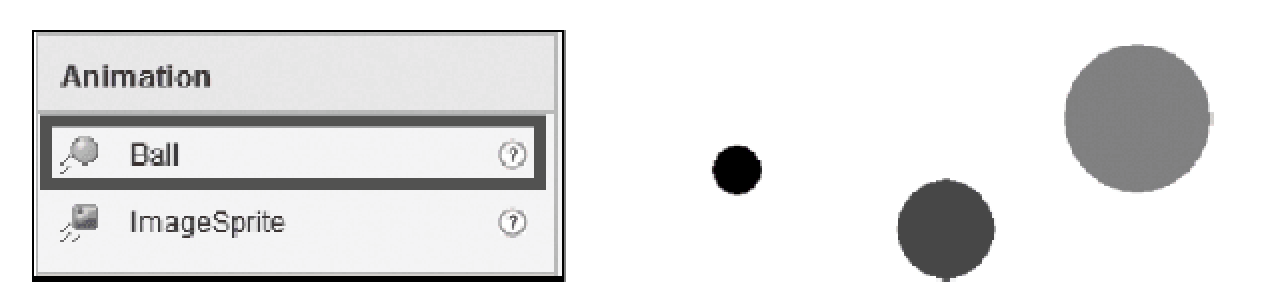


图 6.51　球体控件

球体的属性包括球体大小、颜色、移动频率、移动方向和移动速度等，如表 6.9 所示。

表 6.9　球体的属性

属　性	说　　明	属　性	说　　明
Radius	球的半径	Visible	球体是否可见
PaintColor	球体颜色	Heading	球体移动方向
Enabled	是否启动球体	Speed	球体移动速度
Interval	球体移动频率		

球体与图像精灵具有相同的事件和方法，下面的内容主要介绍与碰撞检测相关的事件和方法。球体中与碰撞检测相关的事件有碰撞事件、不再碰撞事件和触壁事件。

碰撞事件(CollidedWith)是当前图像精灵与其他精灵或画布边缘发生碰撞时产生的事件，该事件的参数 other 是被碰撞精灵的名称，如图 6.52 所示。

不再碰撞事件(NoLongerCollidingWith)是当前精灵与已经产生碰撞的精灵分开时产生的事件，是碰撞事件的对立事件，如图 6.53 所示。当两个精灵产生碰撞时，先发生的是碰撞事件，如果两个精灵被弹开，随即便会产生不再碰撞事件。该事件的参数 other1 是被碰撞精灵的名称。



图 6.52　球体的碰撞事件

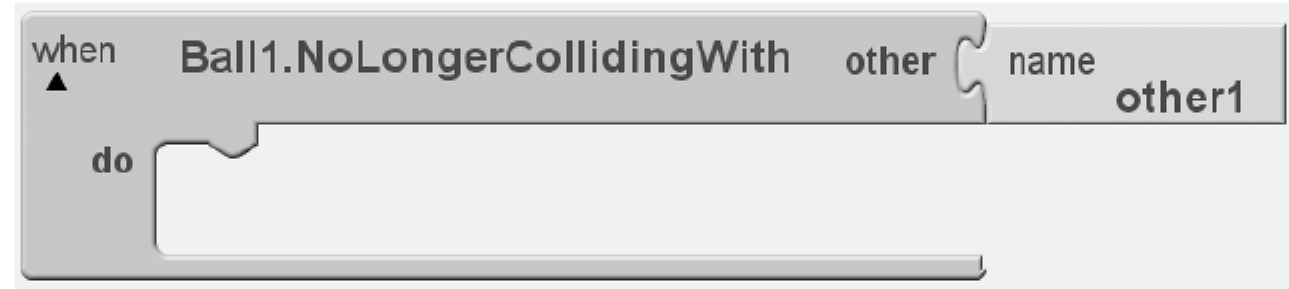


图 6.53　球体的不再碰撞事件

触壁事件(EdgeReached)在图像精灵移动到画布边缘时发生,如图 6.54 所示。当一个精灵运动到画布边缘时,则会产生触壁事件,如果不对此事件做任何处理,精灵则会移动出画布边界;相反,如果在触壁事件中将精灵的运动方向逆转,则精灵又会重新回到画布中。

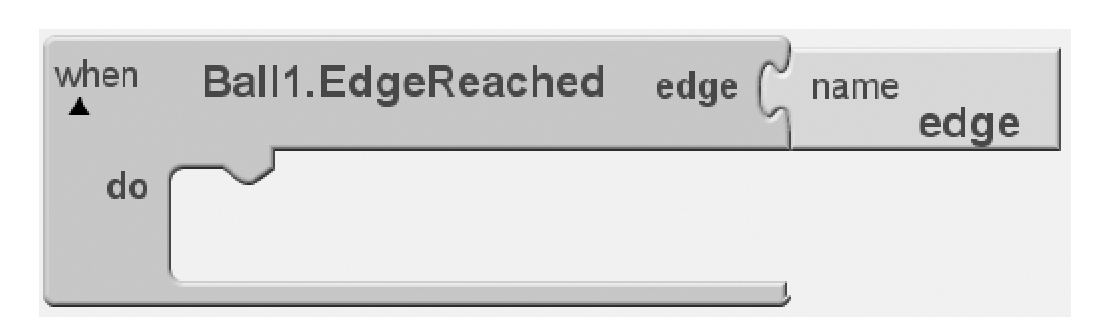


图 6.54　球体的触壁事件

触壁事件的参数 edge 表示所到达的画布边缘,用数字表示画布不同的边缘,其具体的含义如图 6.55 所示。1 表示上方(north),2 表示右上(northeast)、3 表示右侧(east)、4 表示右下(southeast),−1 表示下方(south)、−2 表示左下方(southwest),−3 表示左侧(west)、−4 表示左上方(northwest)。

在球体的方法中,最常用的就是 Bounce 方法,与 EdgeReached 事件配合使用可将球体从画布边缘反弹回画布中。当 EdgeReached 事件检测到碰撞后,调用 Bounce 方法,并将 EdgeReached 事件检测到的碰撞边缘 Edge 参数传递给 Bounce 方法,精灵可以根据此参数进行边缘反弹。边缘检测事件逻辑模块如图 6.56 所示。

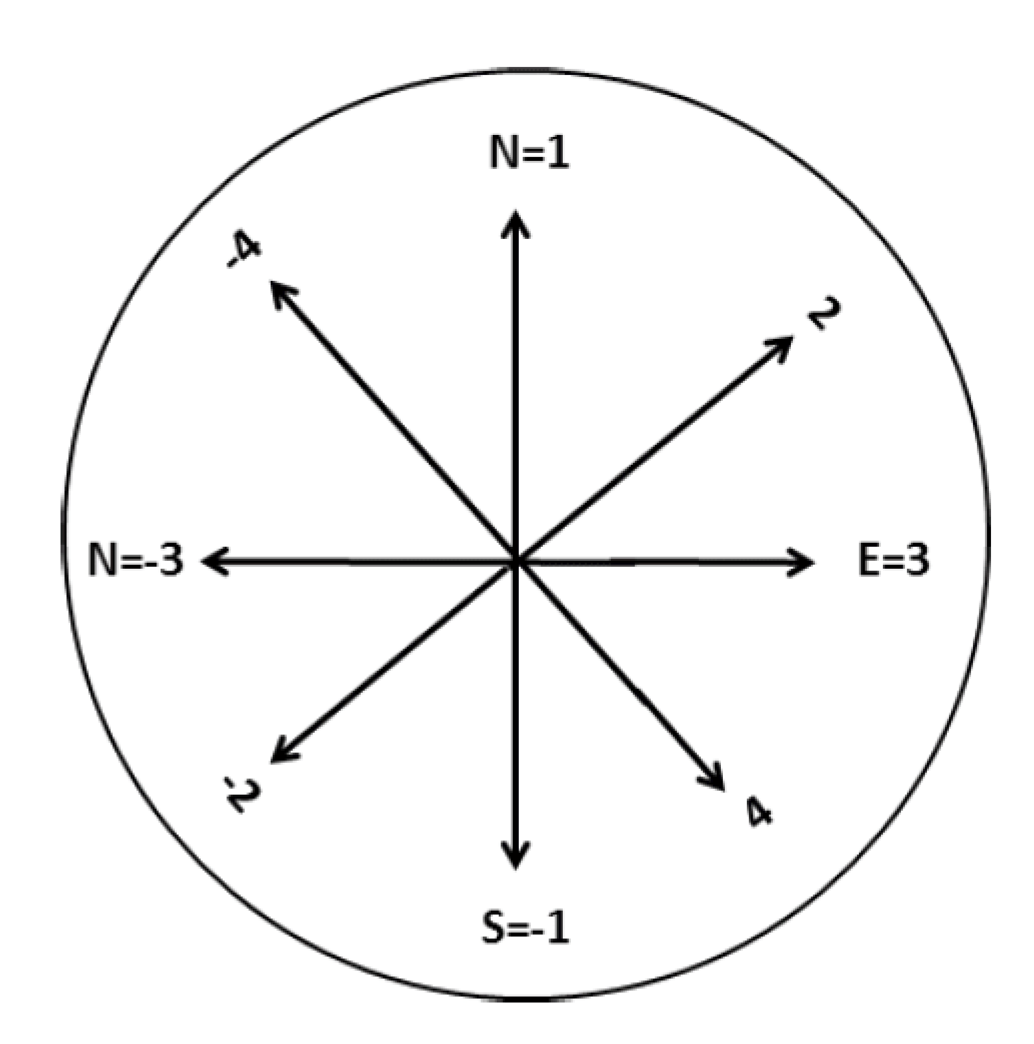


图 6.55　画布边缘信息含义

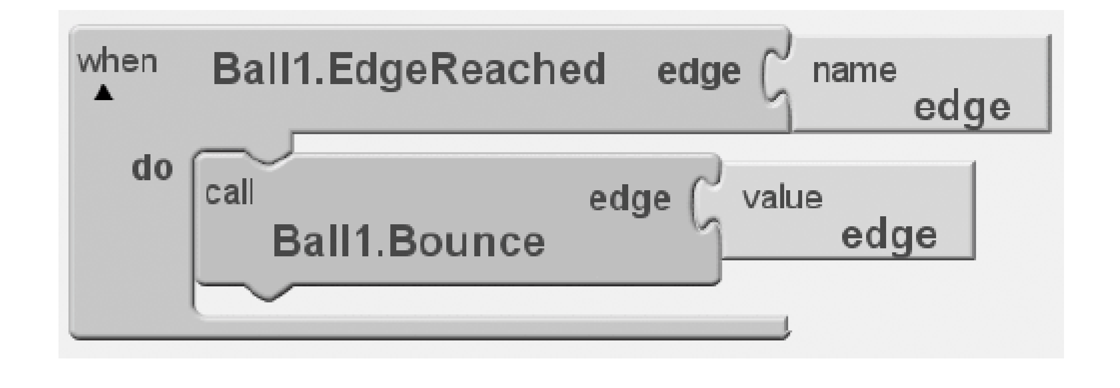


图 6.56　边缘检测事件逻辑模块

6.3.3　方向传感器

在介绍“乒乓球”示例之前,先来说明一下如何使用手机的方向传感器。方向传感器控件(OrientationSensor)是一个非可视化控件,可用来测定手机的三轴角度的变化。界面编辑器中的方向传感器控件如图 6.57 所示。

将手机正面向上地放置在水平桌面上,沿手机屏幕水平方向(左右)定义为 x 轴,水平方向(后前)定义为 y 轴,垂直于手机屏幕的方向(上下)定义为 z 轴,如图 6.58 所示。

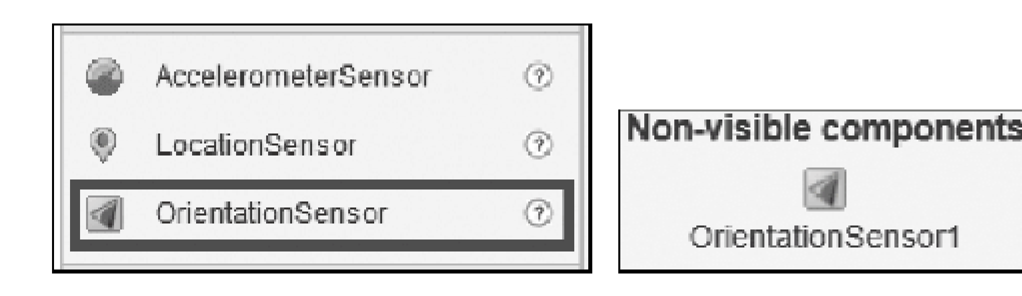


图 6.57 方向传感器

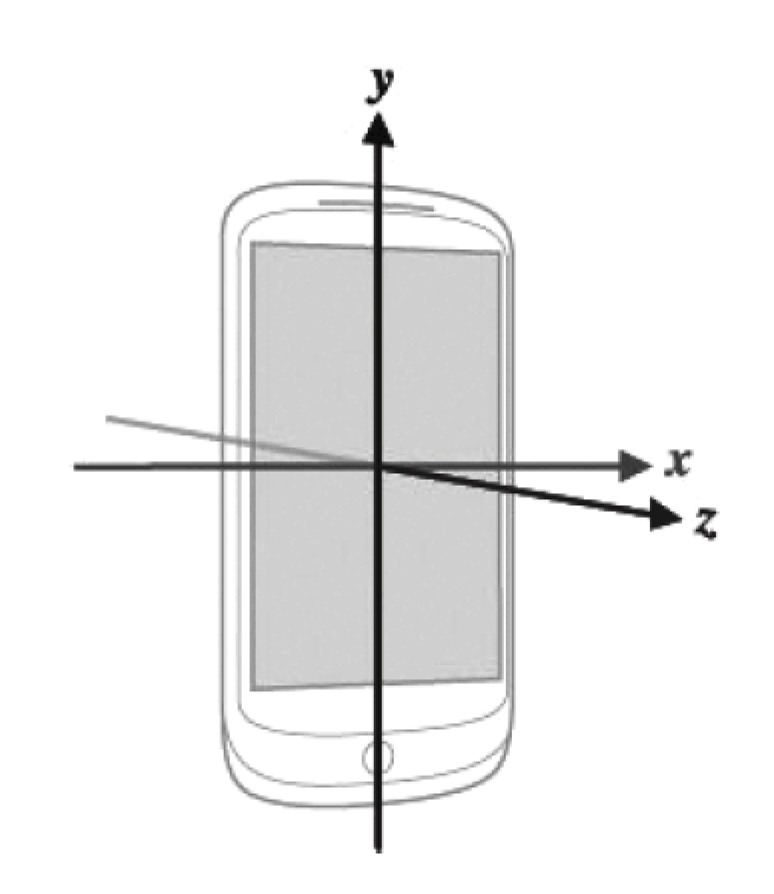


图 6.58 手机三个维度示意图

方向传感器可以反馈三个数据 Roll、Pitch 和 Azimuth。方向传感器的数据单位是度，且有正负之分。

Roll 表示手机 x 轴与水平面的夹角，范围为－90～90。手机处于水平放置时数值为 0，向左侧倾斜时数值由 0 递增到 90，向右倾斜时数值由 0 递减至－90。Pitch 表示手机 y 轴与水平面的夹角，范围为－90～90。手机水平放置时为 0，y 轴正方向朝下倾斜时数值由 0 增加到 90，y 轴正方向朝上倾斜时数值由 0 递减至－90。Azimuth 表示方位，手机水平放置时磁北极和 y 轴的夹角，范围为 0～360。y 轴正方向指向正北时为 0，指向正东为 90，指向正南为 180，指向西东为 270。

方向传感器控件通过相关属性包括方向传感器的启动、方位角和设备翻转大小等，来确定设备的方向改变，如表 6.10 所示。

表 6.10 方向传感器属性

属 性	说 明	属 性	说 明
Available	方向传感器是否可用	Roll	水平转动角度
Enabled	启用方向传感器	Magnitude	倾斜程度
Azimuth	方位角	Angle	倾斜角
Pitch	垂直翻转角度		

Magnitude 表示手机倾斜程度，是一个介于 0 和 1 之间的数。Angle 表示倾斜角大小。

方向传感器只有一个事件 OrientationChanged，在手机方向发生变化时产生，返回 roll、pitch 和 azimuth 数据，如图 6.59 所示。即使静止地放置在水平面上，手机方向传感

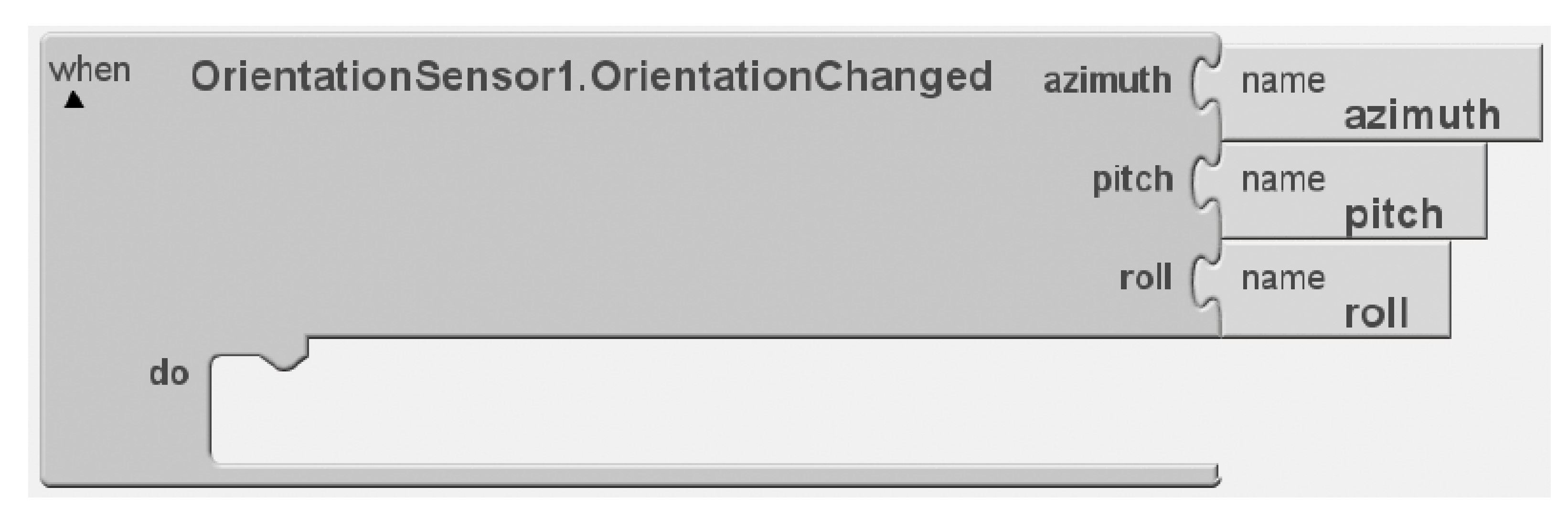


图 6.59 方向传感器的 OrientationChanged 事件

器仍然能够检测到微小的方向变化，所以可以认为OrientationChanged事件是持续发生的事件，事件中的处理过程务必要简单、高效。

下面通过Orientation示例，可令读者使用自己的手机感受如何使用方向传感器，以及分析Roll、Pitch和Azimuth数据的具体含义。Orientation示例的运行界面如图6.60所示，界面设计示意图如图6.61所示。

Orientation示例的全部逻辑模块如图6.62所示。

Orientation
方向传感器三轴姿态数值
X轴: Pitch
-0.44107
Y轴: Roll
-2.75778
Z轴: Azimuth
79.32386

图 6.60　Orientation 示例的运行界面

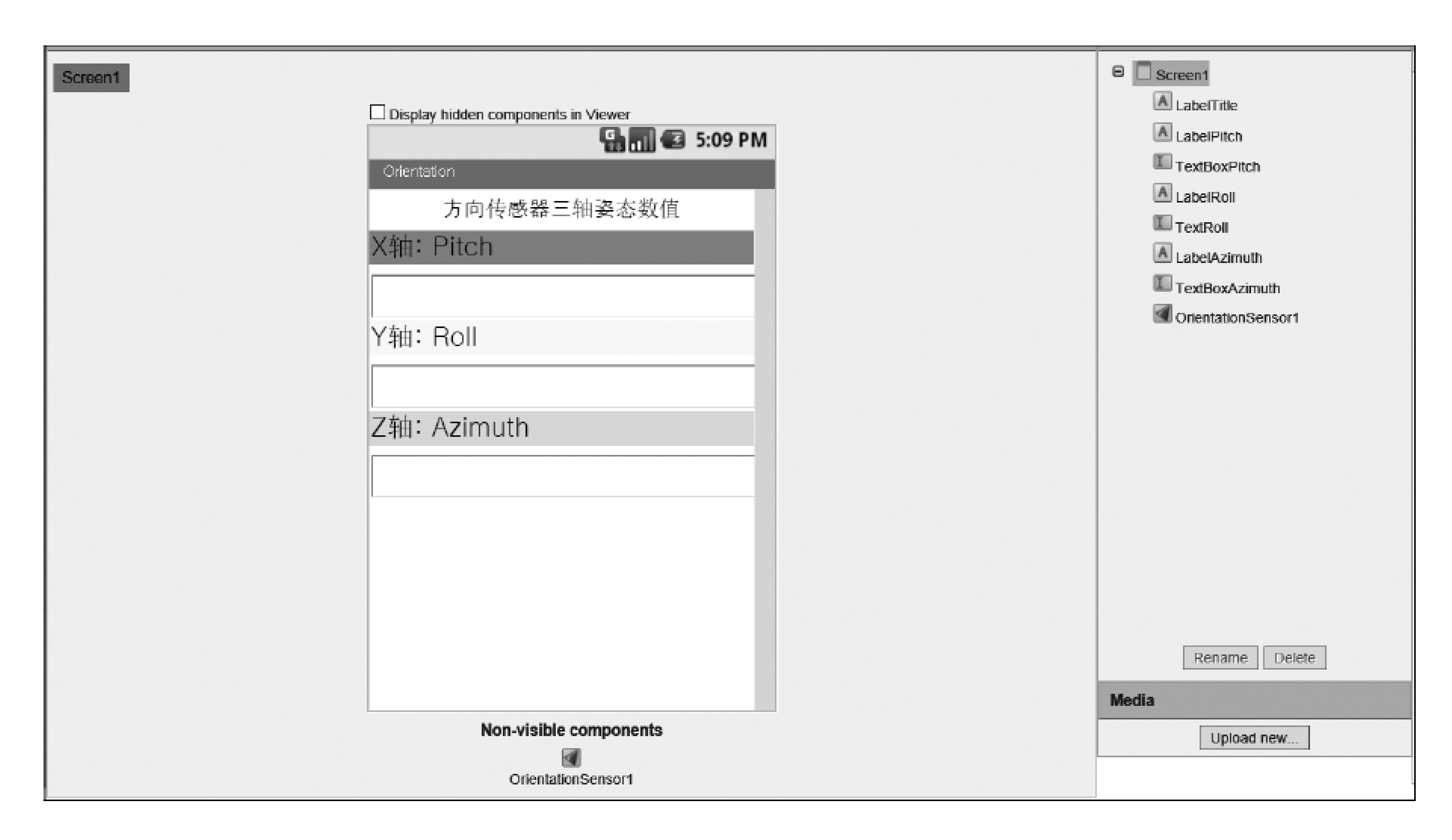


图 6.61　Orientation 示例界面设计图

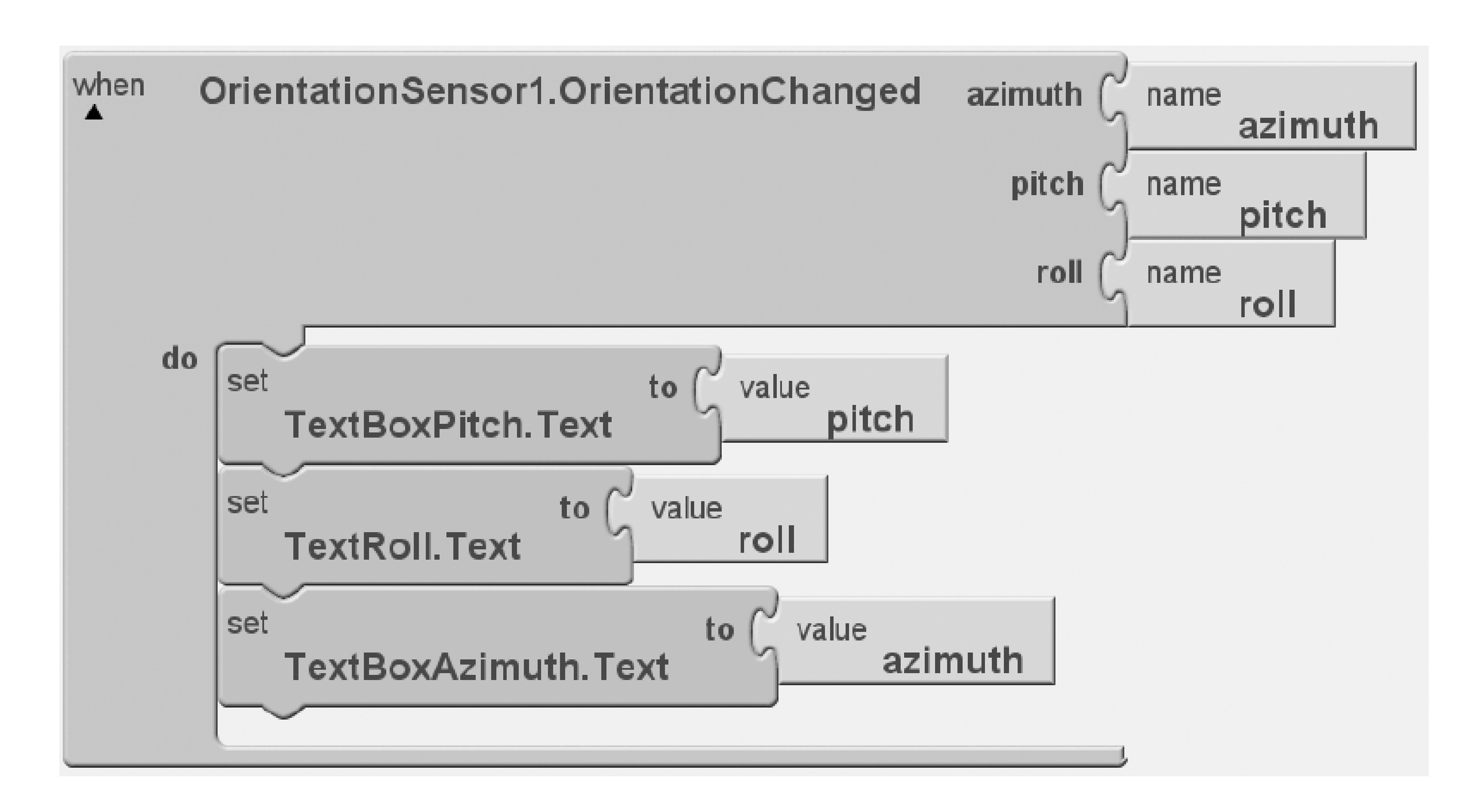


图 6.62　Orientation 示例的全部逻辑模块

6.3.4　示例——乒乓球

本节介绍一个经典的小游戏“乒乓球”，游戏中的小球会向画面下方以一定角度掉落，通过控制画面下方的球板，阻止小球落到地面上。小球碰到球板后会弹起，每次接触

球板都会得 1 分。游戏中可以使用触碰的方法控制球板,也可以使用方向传感器控制球板。

Pong 示例的运行界面如图 6.63 所示。



图 6.63 Pong 示例的运行界面

Pong 示例使用的控件有球体、图像精灵、画布、方向传感器、音效播放器,以及常见的按钮、标签和复选框等控件。Pong 示例的界面设计如图 6.64 所示。

将所有游戏需要的资源文件上传到 App Inventor 中,包括 Play 按钮的背景图片 PlayButton.png、Reset 按钮的背景图片 ResetButton.png、画布的背景图片 background.png、球板图像精灵的图片 Paddle.png 和失败声效文件 DingDong.mp3。

游戏是纵向设计的,这里关闭手机的屏幕旋转功能,将 Screen1 的 ScreenOrientation 属性设置为 Portrait。将音效播放器的 Source 属性设置为 DingDong.mp3。还要将球体的初始位置设置为(20,20),即属性 X 为 20,Y 属性为 20,球体的半径(Radius)也设置为 20。

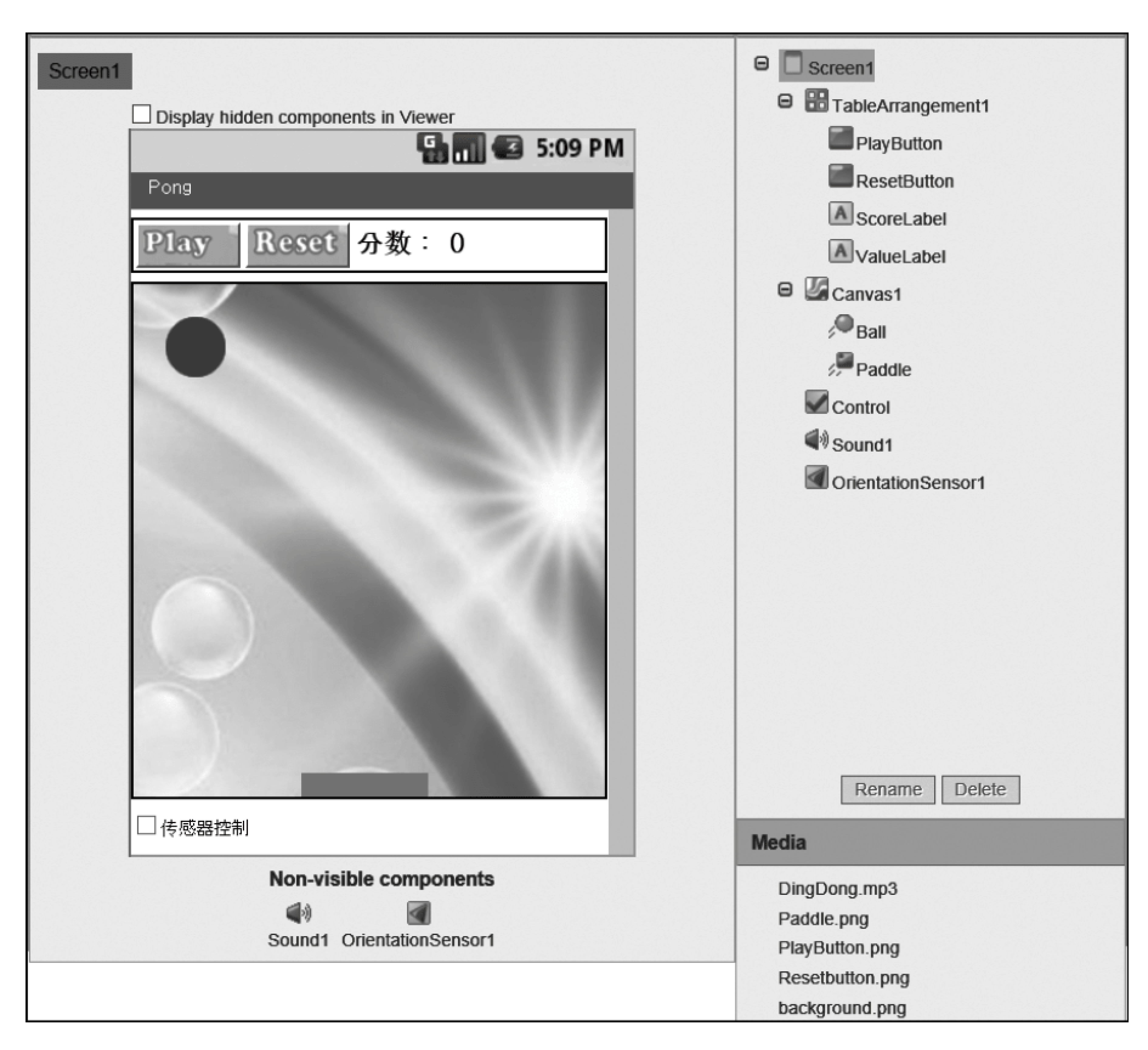


图 6.64 Pong 示例的界面设计图

在完成界面设计器的工作后,开始程序的逻辑设计部分。如图 6.65 所示,首先在屏幕页的 Initialize 事件中,将球体 Enabled 属性设置为 false,避免球体在没有正式开始游戏前移动。还要将方向传感器的 Enabled 属性设置为 false,在用户手动选择"传感器控制"复选框后,才可以启动传感器。

"传感器控制"复选框的Changed事件的逻辑如图6.66所示。在每次复选框修改事件中,判断复选框的状态,如果复选框被选中,则启动方向传感器,否则就关闭方向传感器。

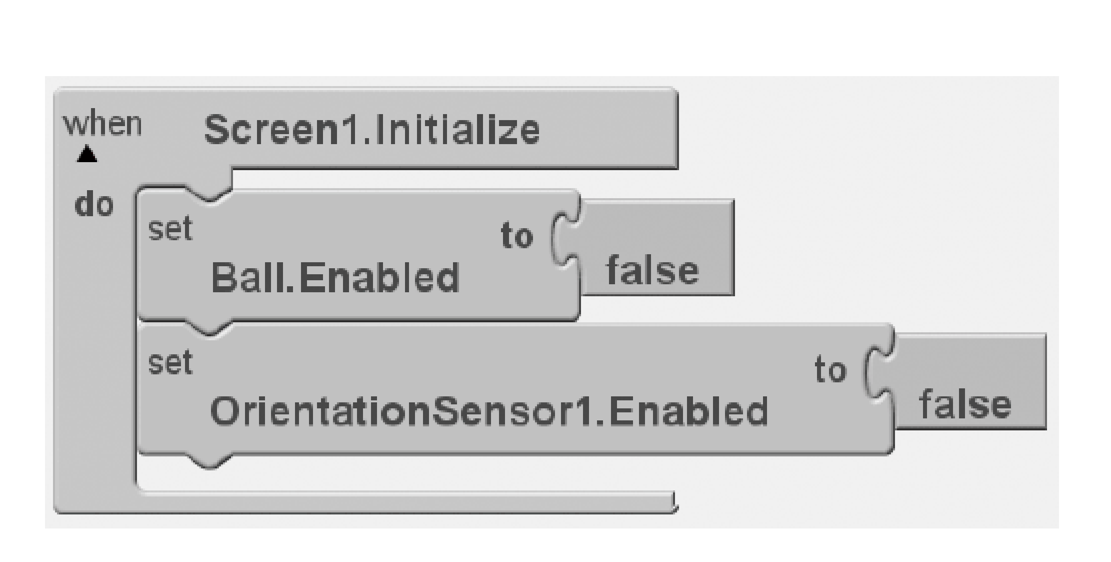


图6.65　屏幕页的Initialize事件

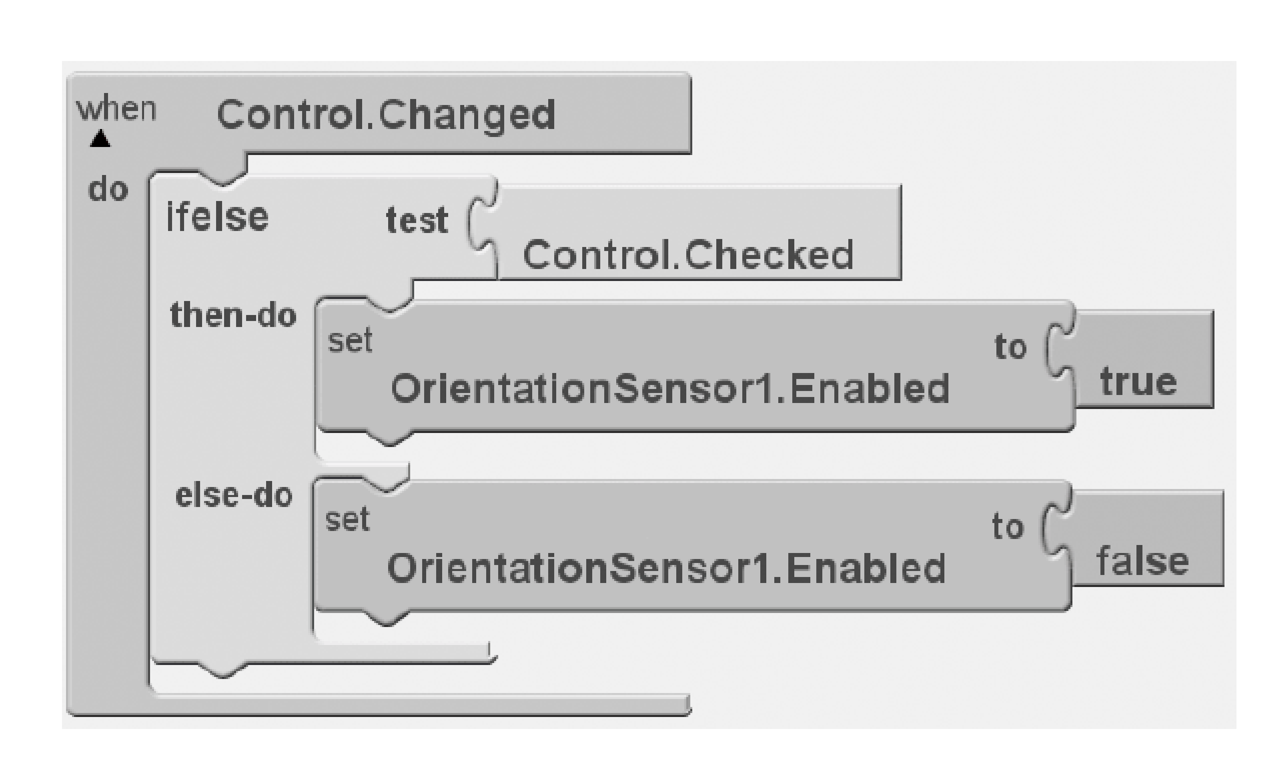


图6.66　复选框的Changed事件

使用方向传感器控制球板是一个不错的选择,这样用户就可以不接触手机屏幕,利用手机的倾斜角度控制球板的移动方向,但需要注意传感器控制球板的敏感程度。在用户操作手机过程中,即使用户水平放置手机,也难以避免产生微小的倾斜,但此时用户并不希望移动球板。方向传感器的属性roll表示手机水平方向的倾斜角度,这里设定一个门限值,只有roll超出门限值时,才可以控制球板,否则球板将在原位置上不移动。设置门限值的好处是避免球板过于敏感,不按照用户的意图进行移动。Pong示例中这个门限值是4,在图6.67中,条件"abs(roll)>4"就是门限值的具体实现,如果roll的值小于4,则无法移动球板。

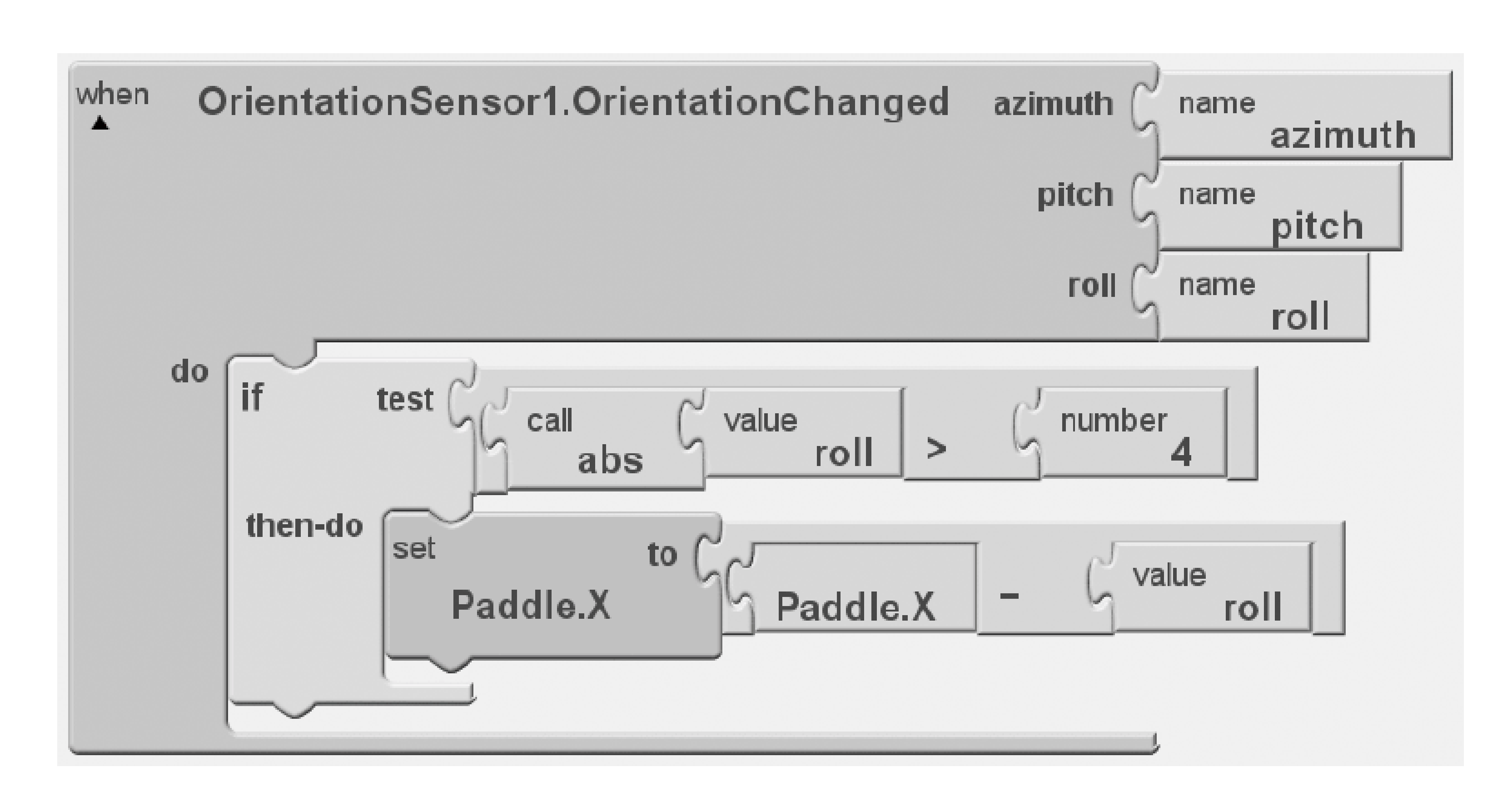


图6.67　方向传感器的OrientationChanged事件

在方向传感器的OrientationChanged事件中,roll的方向决定了球板的移动方向。roll为正值,球板的X坐标值减小,球板向左侧移动;roll为负值,球板的X坐标值增加,球板向右侧移动。

除了可以使用方向传感器控制球板以外,还可以使用在屏幕上滑动手指来控制球板,这种方式似乎更加普遍。在图像精灵Paddle的Dragged事件中,修改图像精灵Paddle的属性X值,就可以移动球板,如图6.68所示。

在球板的拖曳过程中,用户触碰屏幕点,本意都是希望球板的中心点到达触碰点。currentX是触碰点的X坐标,在此基础上减去40,就可以将球板精灵的中心点移动到触

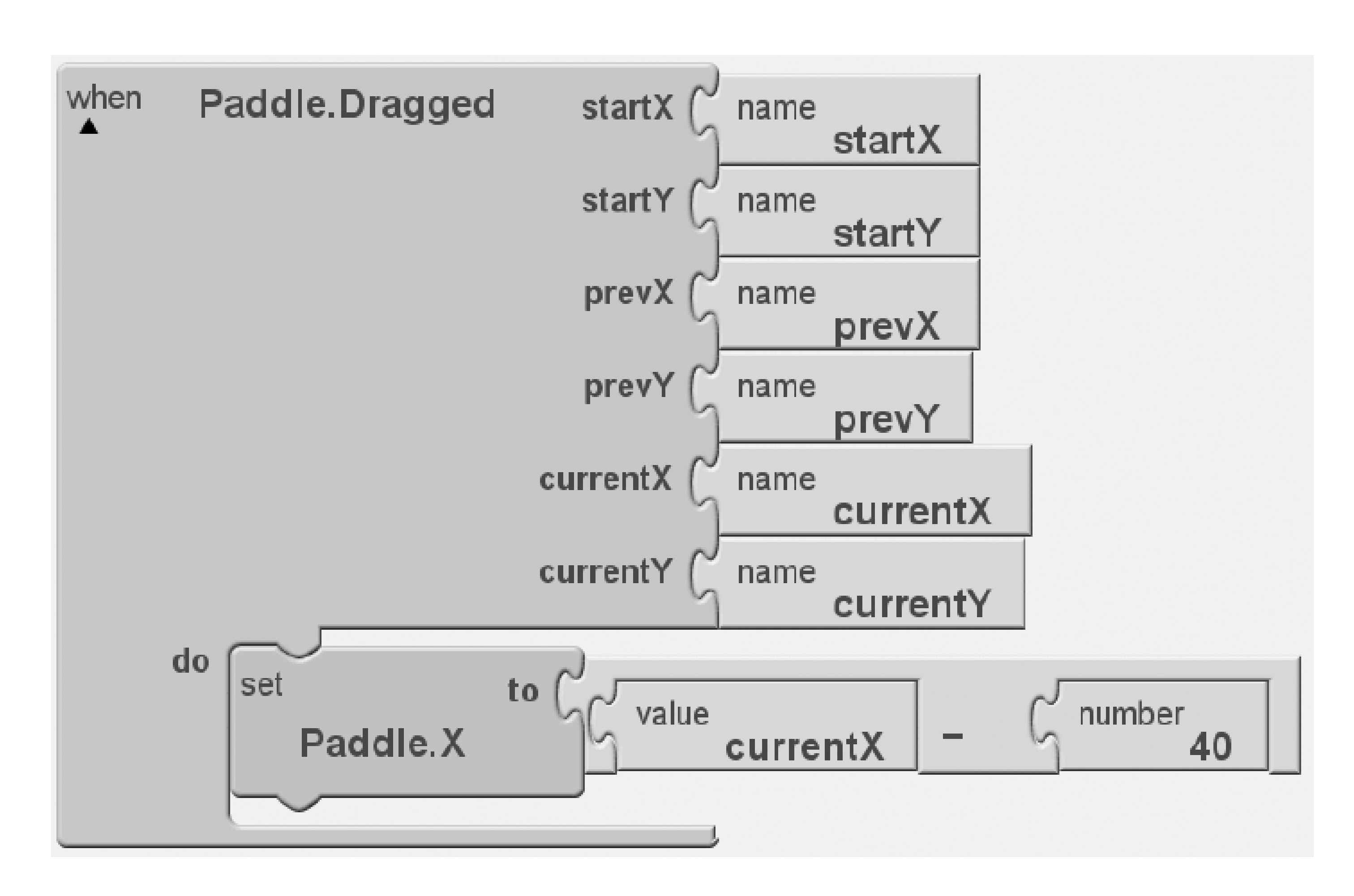


图 6.68 图像精灵 Paddle 的 Dragged 事件

碰点上来。

开始游戏要单击 Play 按钮,重置游戏要单击 Reset 按钮,这两个按钮的单击事件的逻辑模块如图 6.69 所示。在 PlayButton 的单击事件中,首先调用自定义的函数模块 initGame,初始化球体控件的基本参数,然后使球体控件运动(Enabled 设置为 true)。在 ResetButton 的单击事件中,先让球体控件停止运动(Enabled 设置为 false),再调用自定义的函数模块 initGame。

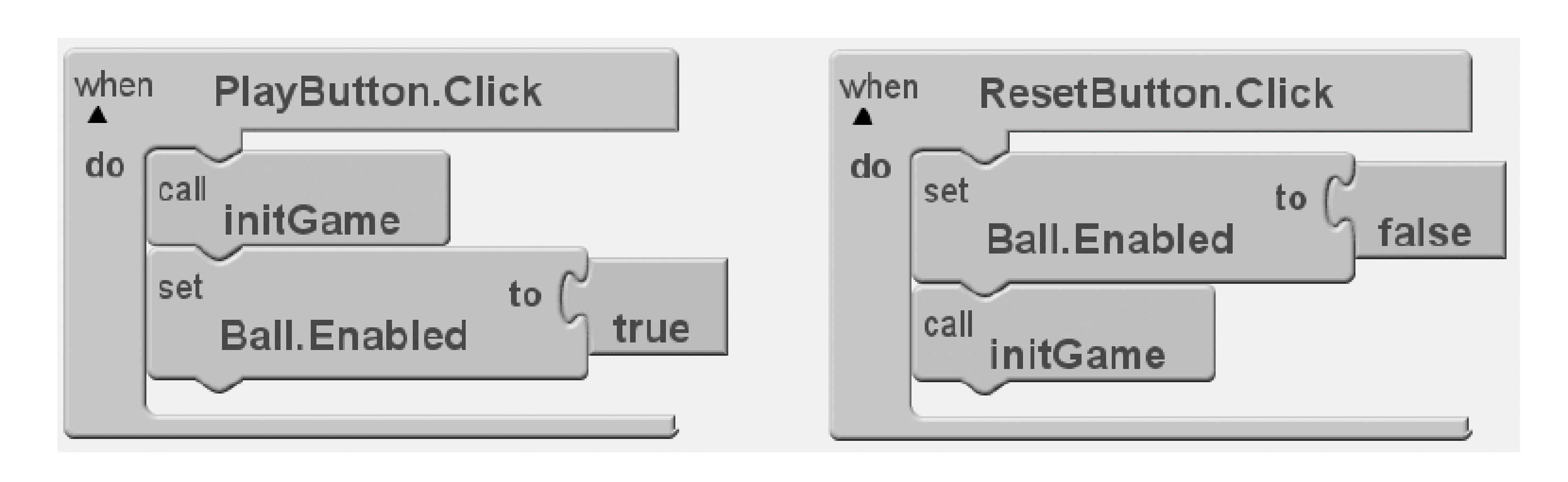


图 6.69 按钮单击事件

自定义函数 initGame 用来初始化球体控件的位置、移动速度、移动方向,并清空分数标签控件 ValueLabel 的文字属性 Text。函数 initGame 将球体的位置设置到(20,20),球体的初始速度在 8～15 之间随机选择,球体的移动方向在 0～360 之间随机选择,如图 6.70 所示。

球体在运动时碰撞到画布边缘,将产生球体的 EdgeReached 事件。根据游戏规则,球体碰到上方、左侧和右侧的边缘时,球体会按照一定角度进行反弹;如果球体碰到下方的边缘,游戏会结束,球体会停止在触碰点上。因此在球体的 EdgeReached 事件中,首先要检测参数 edge 的值,当 edge 为-1 时,表示小球触碰到画布的下方边缘,则将球体的 Enabled 属性设置为 false,使小球停止移动,并播放表示游戏结束的音效;当 edge 为其他值时,调用球体的方法 Bounce,根据参数 edge 进行反弹。EdgeReached 事件的内部模块如图 6.71 所示。

当球体运动碰撞到球板时,球体应当按照一定角度进行反弹,此时将产生球体的 CollidedWith 事件。在球体 CollidedWith 事件中,参数 other 表示与球体碰撞的精灵,因

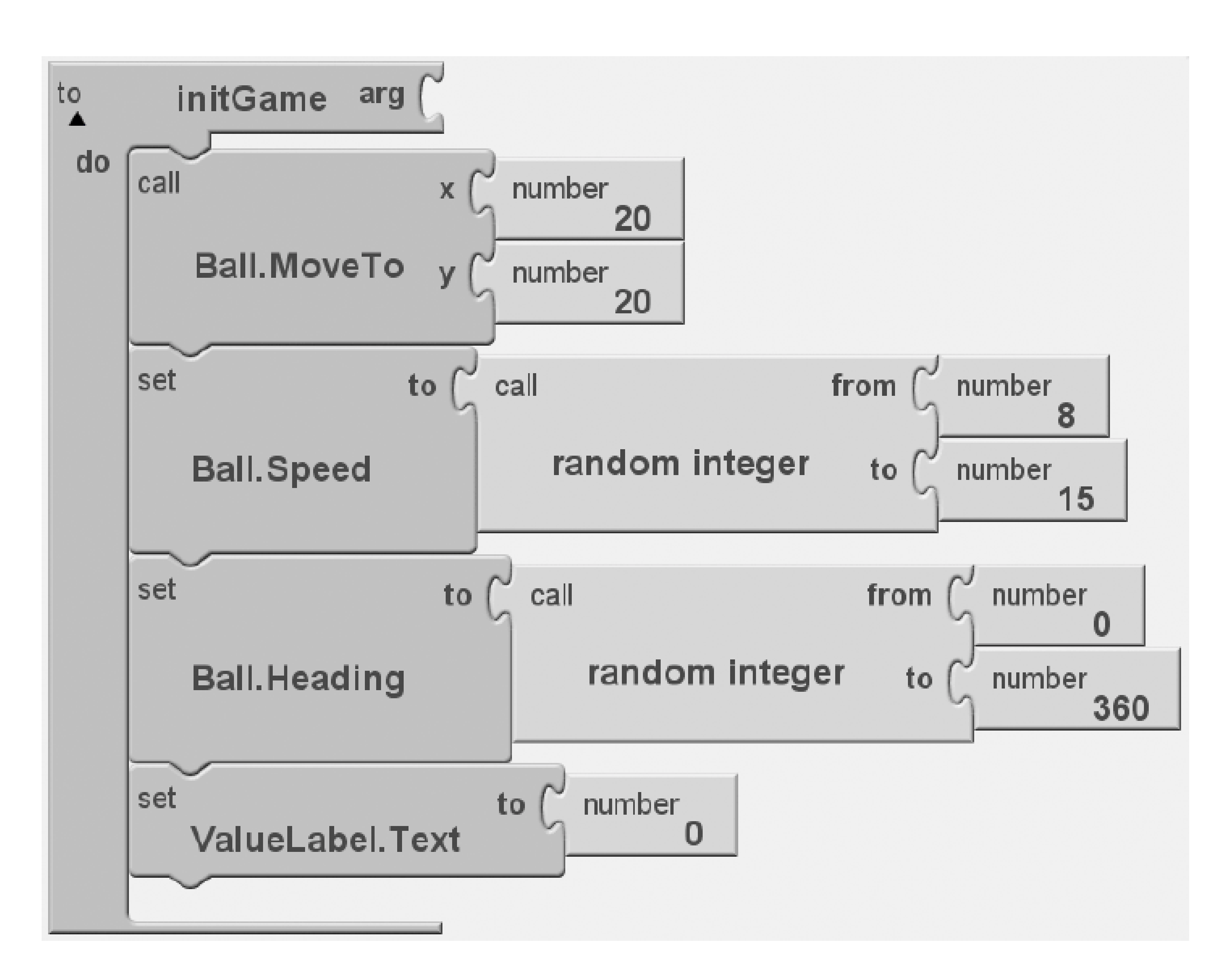


图 6.70　自定义函数 initGame

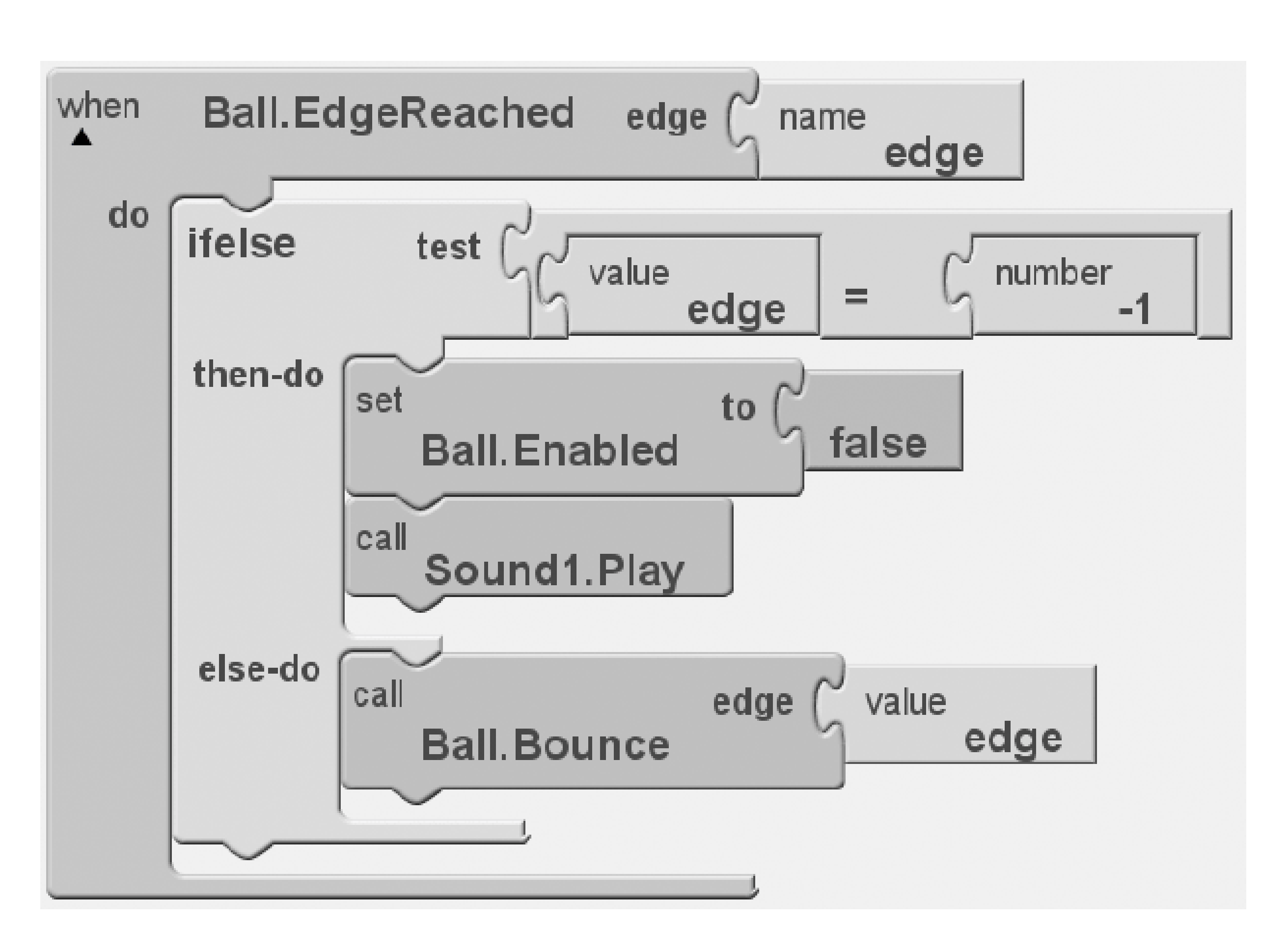


图 6.71　球体的 EdgeReached 事件

为画布上只有球体和球板，因此在每次碰撞事件中，other 一定是球板，但这里的参数 other 并没有被使用。在 CollidedWith 事件中，首先用 360 减去球的运动方向，表示球体碰撞后的物理反弹，然后将记分的标签 ValueLabel 的数值增加 1，如图 6.72 所示。

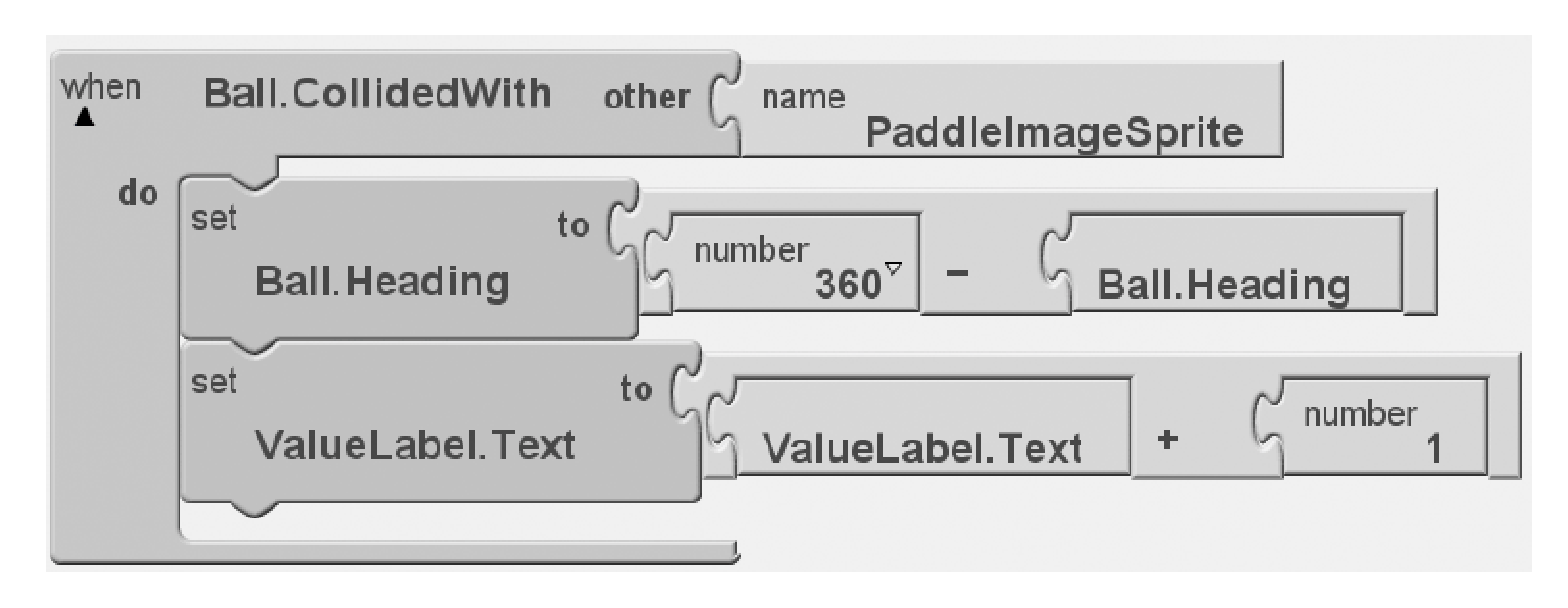


图 6.72　球体的 CollidedWith 事件

Pong 示例的全部逻辑模块如图 6.73 所示。

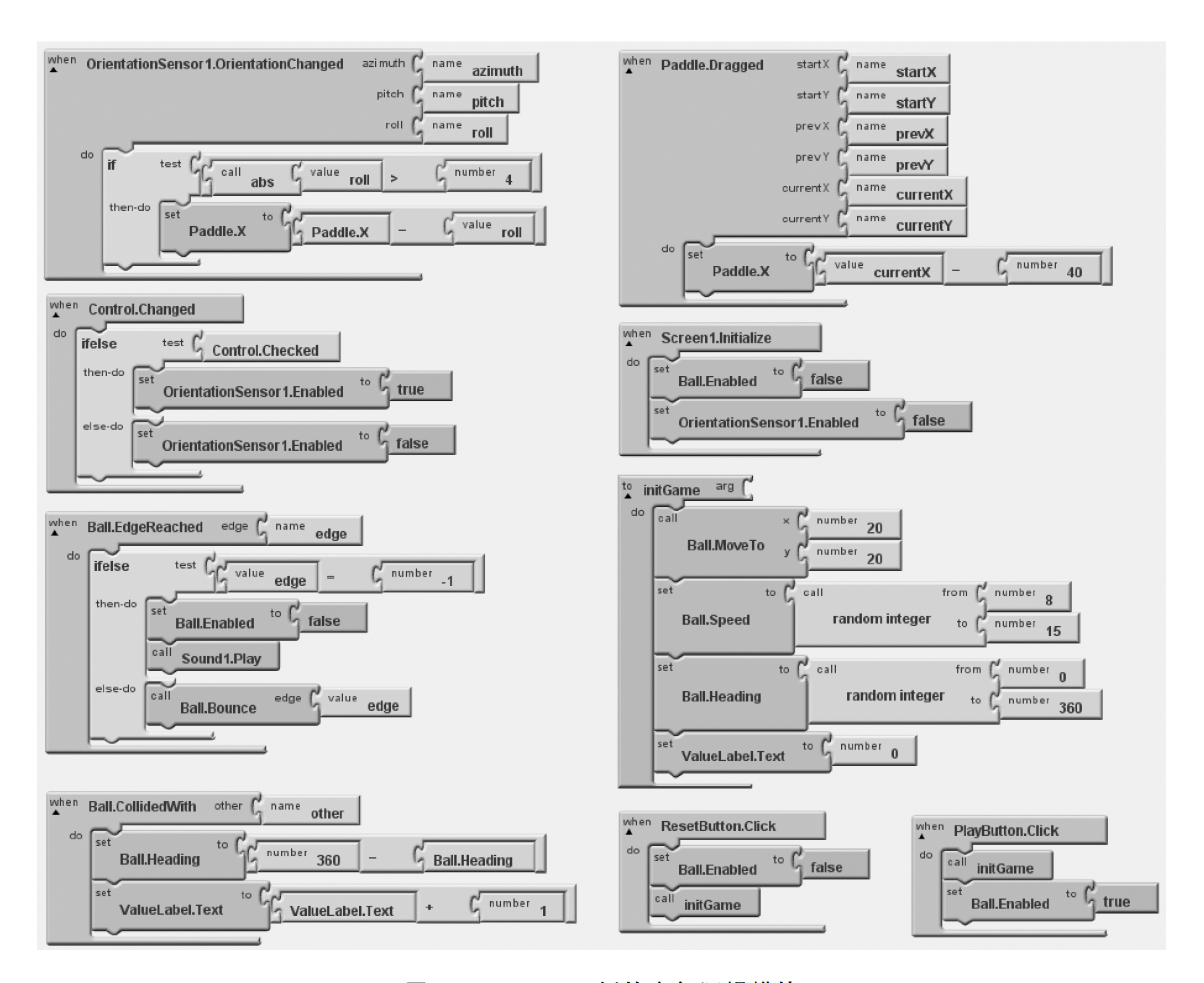


图 6.73 Pong 示例的全部逻辑模块

习　　题

1. 说明图像精灵和球体的不同和相同之处。

2. 编写实现“新画板”功能，当手指单击屏幕的任意两个位置时，以这两个位置作为矩形的左上角和右下角绘制矩形图案，可以选择三种不同的颜色，并可随时清空画布内容。

3. 编写实现“石头剪子布”游戏，实现人与手机的石头剪刀布游戏功能，每比较一次，输出获胜一方的信息。

4. 编程实现“打气球”游戏。在游戏开始后，画面上随机出现 10 个球，这些气球是会任意飘动的，用户要在最短的时间内击破所有气球，完成游戏后显示玩家所使用的时间。

第 7 章



数据存储与访问

App Inventor 提供非常简单的数据存储和读取方法。通过本章的学习可以帮助读者掌握 App Inventor 中实现数据存储和读取的控件 TinyDB。

本章学习目标：

- 掌握 TinyDB 数据存储与访问方法；
- 掌握 TinyDB 数据更新和删除方法。

7.1 基础功能

在开发应用程序过程中，经常会有一些数据需要长时间地存储起来，以便在需要时可以立即获取到这些数据。App Inventor 提供了简单的数据存取控件 TinyDB，是一种基于 NVP(Name/Value Pair，标签/值对)的数据存储方式，用户无须了解数据的存储细节，就可以根据标签在数据库中保存、读取数据。

这里所说的 NVP 是使用“标签”进行数据存储的模式，每个数据对应一个标签。例如，用户需要保存一个叫作 Lucy 的名字，只需要存入(name，Lucy)就可以了，这里的 name 是标签，值是 Lucy。在从数据库中获取这个数据时，只要提供标签 name，就能在数据库中找到需要的数据 Lucy。当然，用户也可以保存(1，HanMeimei)、(2，LiLei)、(ID001，18600001111)和(ID002，18600002222)这样的数据。

TinyDB 控件属于非可视化控件，由于其广泛的适用性和简单的操作方法，在应用程序中使用得非常广泛。TinyDB 控件的添加方式依旧是将其拖入屏幕区，控件将显示在屏幕下方的非可视控件区域内，如图 7.1 所示。

由于 TinyDB 控件是功能性的非可视化控件，因此这个控件没有任何可编辑属性和触发事件，仅支持数据存储(StoreValue)和数据读取(GetValue)两个方法，如图 7.2 所示。



图 7.1 TinyDB 控件



图 7.2 数据存储和数据读取方法

StoreValue方法的功能是将数据保存到数据库中，参数tag是标签，参数valueToStore是被存储的数据，被保存的数据既可以是字符串，也可以是列表。在图7.3中，分别使用列表和字符串方式，将数据Tom、boy、20保存到数据库中。

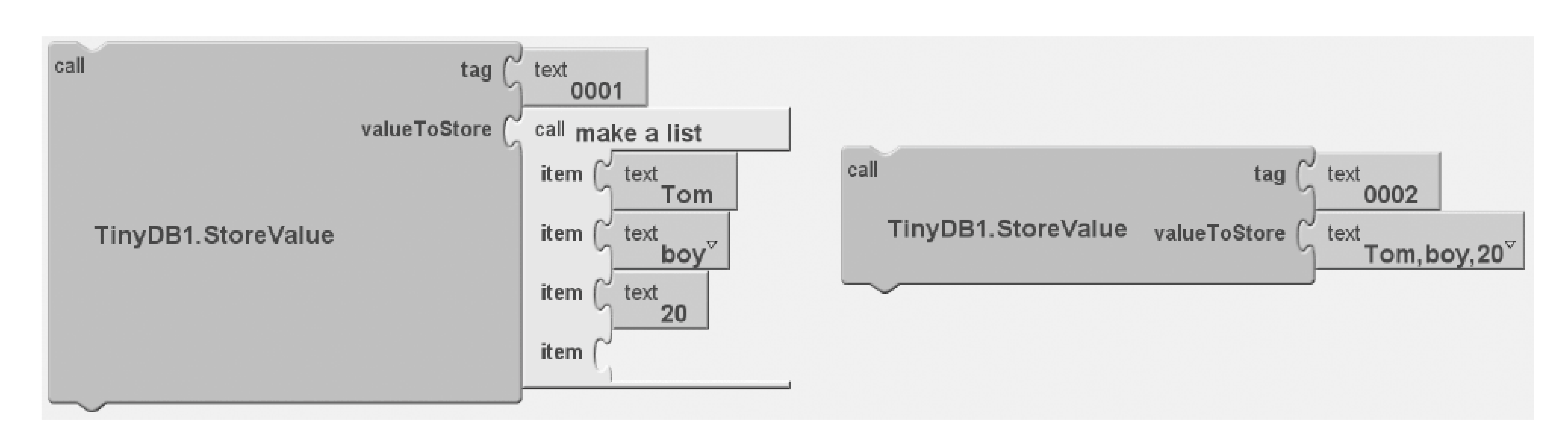


图7.3　数据存储示例

GetValue方法的功能是根据标签获取数据库中的数据，参数tag是标签。在图7.4中，GetValue方法的tag参数是0001，因此根据标签0001在数据库中进行搜索，如果知道匹配的标签，可以返回标签对应的数据；如果没有找到匹配的标签，则返回一个空的字符串。因此，要检验某个标签下是否有存储数据，只要检测返回值是否为空即可。



图7.4　数据提取示例

下面通过DataRecorder示例介绍TinyDB控件的数据存储和数据提取功能。DataRecorder示例的运行界面如图7.5所示。

在显示"数据内容"的文本框中输入要保存的数据，单击"写入数据"按钮就可以将数据存储在数据库中，在"数据库"标签下方以(标签，数据)的形式显示，并在"当前条目数"中显示数据库中的数据总量。

用户在显示"序号"的文本框中填入标签，单击"读取数据"按钮，就可以从数据库中提取到刚刚存储的数据。本示例的所有标签从1开始自动递增，例如1、2、3、4、…如果输入的标签在数据库中不存在，则在"数据值"标签中显示"Not Found"。

如图7.6所示为DataRecorder示例界面示意图。为了让示例更为简单，在界面编辑器中将两个文本框控件的NumbersOnly属性设为true，限定只能输入数字。



图7.5　DataRecorder示例的运行界面

在模块编辑器中，首先定义全局变量Tag，用来表示数据存储的标签，如图7.7所示。

接下来开始设计数据存储的过程。当用户单击"写入数据"按钮时，将文本框内的数据存入数据库，因此，需要给数据自动指定一个标签。

在DataRecorder示例中，在存储数据前，先要递增全局变量Tag，目的是为了避免存储在数据库中的标签重复。递增全局变量Tag和存储数据的模块如图7.8所示。



图 7.6　DataRecorder 示例界面示意图

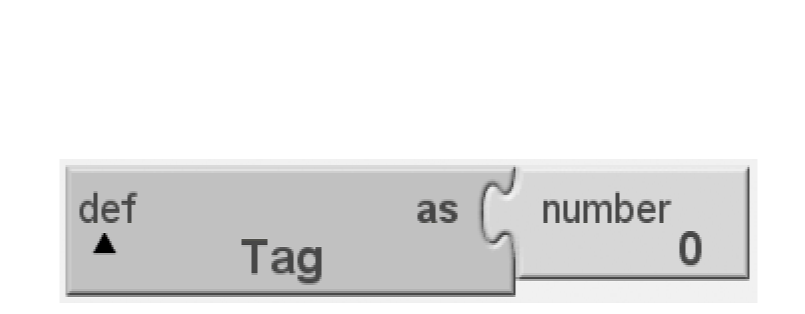


图 7.7　自定义全局变量 Tag

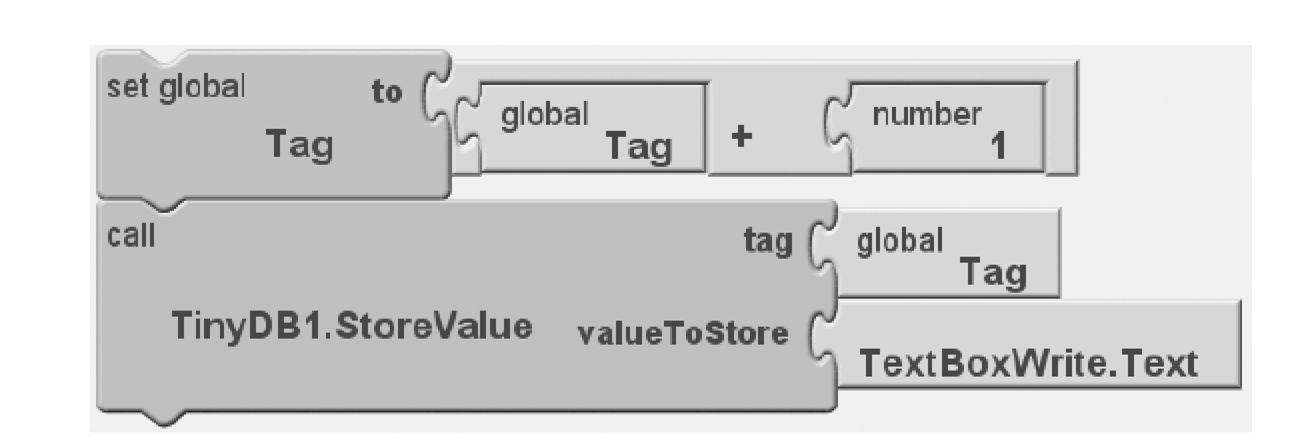


图 7.8　数据存储模块

在完成数据存储过程后，需要在界面上显示存储结果，并清空之前的数据内容，以便用户进行下一次输入。调用 make text 方法可以显示形式为(1,1010101)的数据，如图 7.9 所示。

在用户每次单击 ButtonWrite 按钮后，需要判断文本框是否为空。如果结果为空，则表示用户并没有输入内容，则忽略用户的按钮单击行为。如果结果不为空，则执行数据存储和显示更新功能。ButtonWrite 按钮的单击事件如图 7.10 所示。

数据读取模块如图 7.11 所示，在槽 tag 上拼接上用户在文本框 TextBoxRead 中输入的标签，通过调用 TinyDB1 控件的 GetValue 方法获得数据库中的数据。

最后考虑数据访问部分的逻辑：当单击“读取数据”按钮时，首先判定文本框内的内容是否为空，如果为空则视为误操作，忽略此次运行结果。如果文本框的内容不为空，则将文本框内的内容作为标签在数据库中进行查找。查找完成之后，首先判定查找结果是否为空，如果为空则意味着数据库内的此标签下没有数据存储，即此编号的数据不存在，

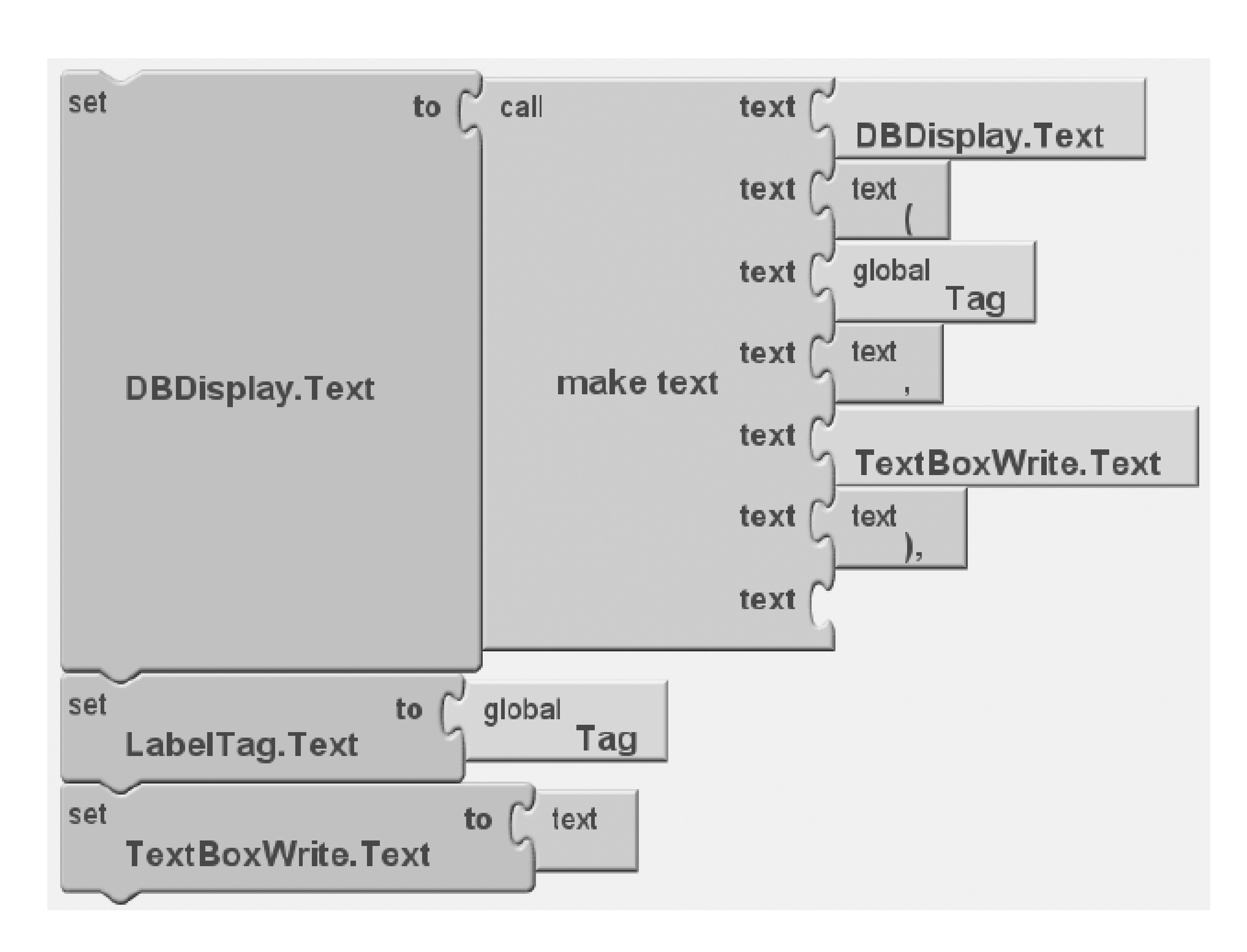


图 7.9　数据存储后的显示更新模块

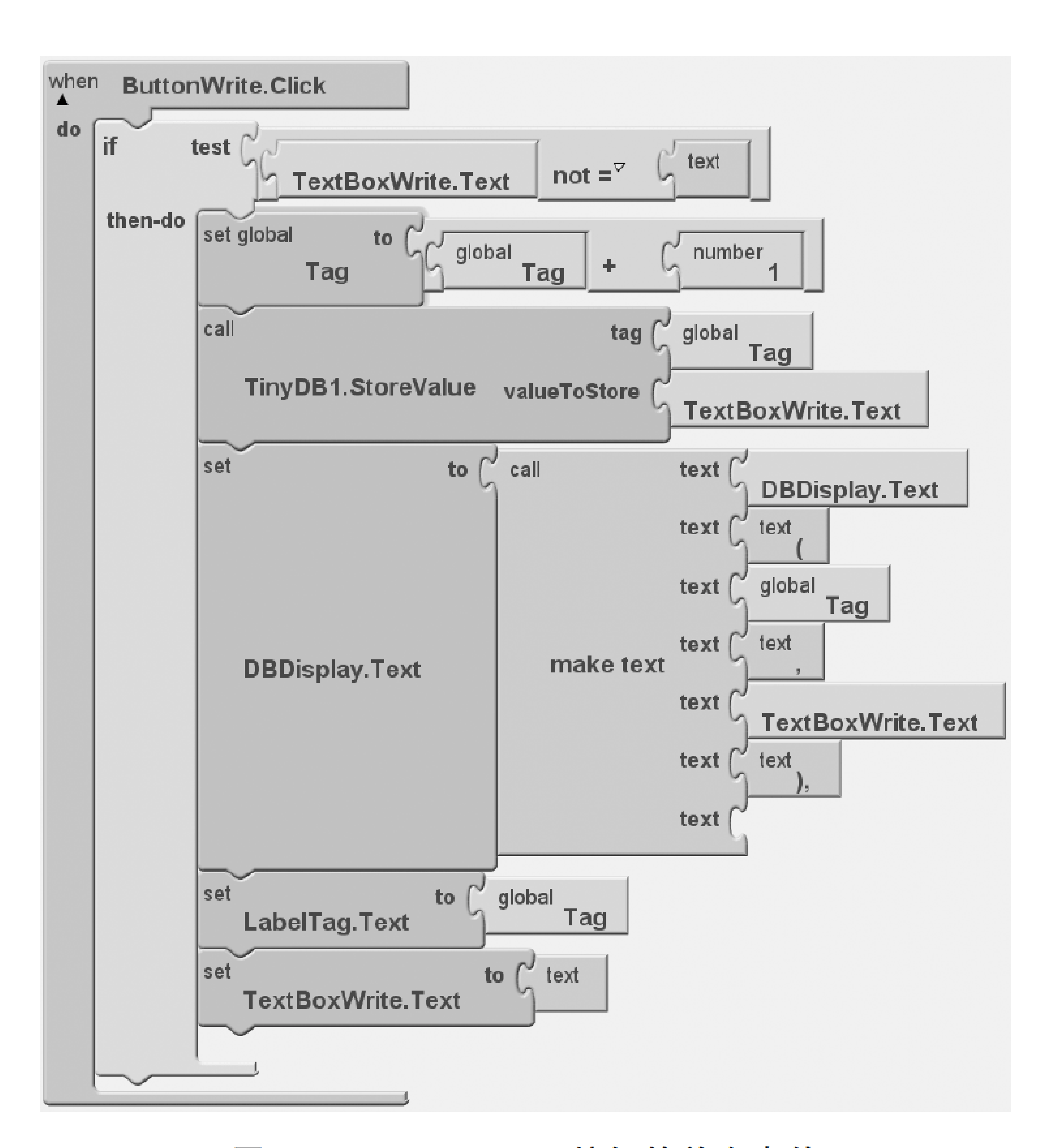


图 7.10　ButtonWrite 按钮的单击事件

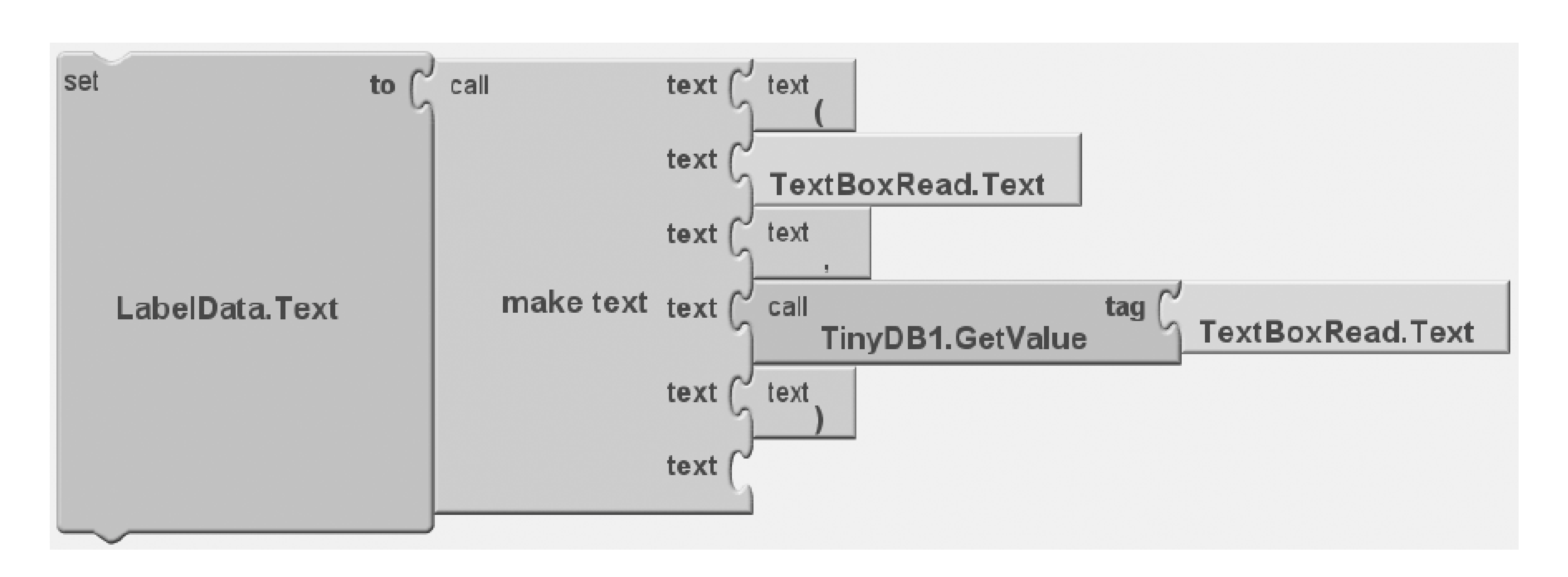


图 7.11　数据读取模块

在此情况下对用户做出“Error!”的提醒；如果数据不为空，则将提取的数据在 LabelData 文本框内进行显示。不管提取结果是否为空，都将输入用的文本框清空，以待下一次输入。

用户每次单击 ButtonRead 按钮后，首先判断文本框是否为空和数据库中是否存有有效数据，如果任何一个条件为 false，则在界面上标签 LabelData 显示“Not Found”；否则将数据库中的返回结果显示在标签 LabelData 中。ButtonRead 按钮的单击事件如图 7.12 所示。

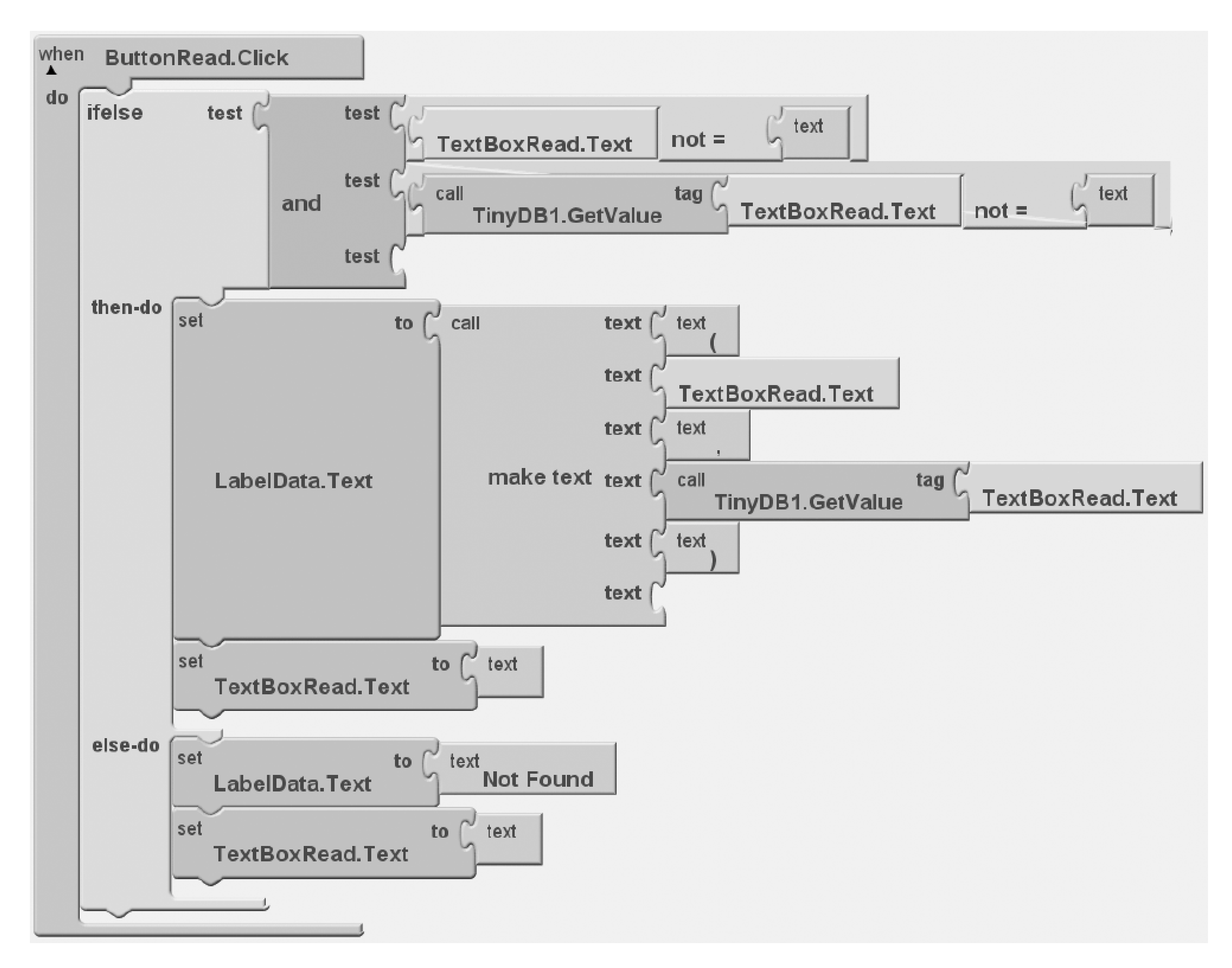


图 7.12 ButtonRead 按钮的单击事件

DataRecorder 示例的完整逻辑模块如图 7.13 所示。

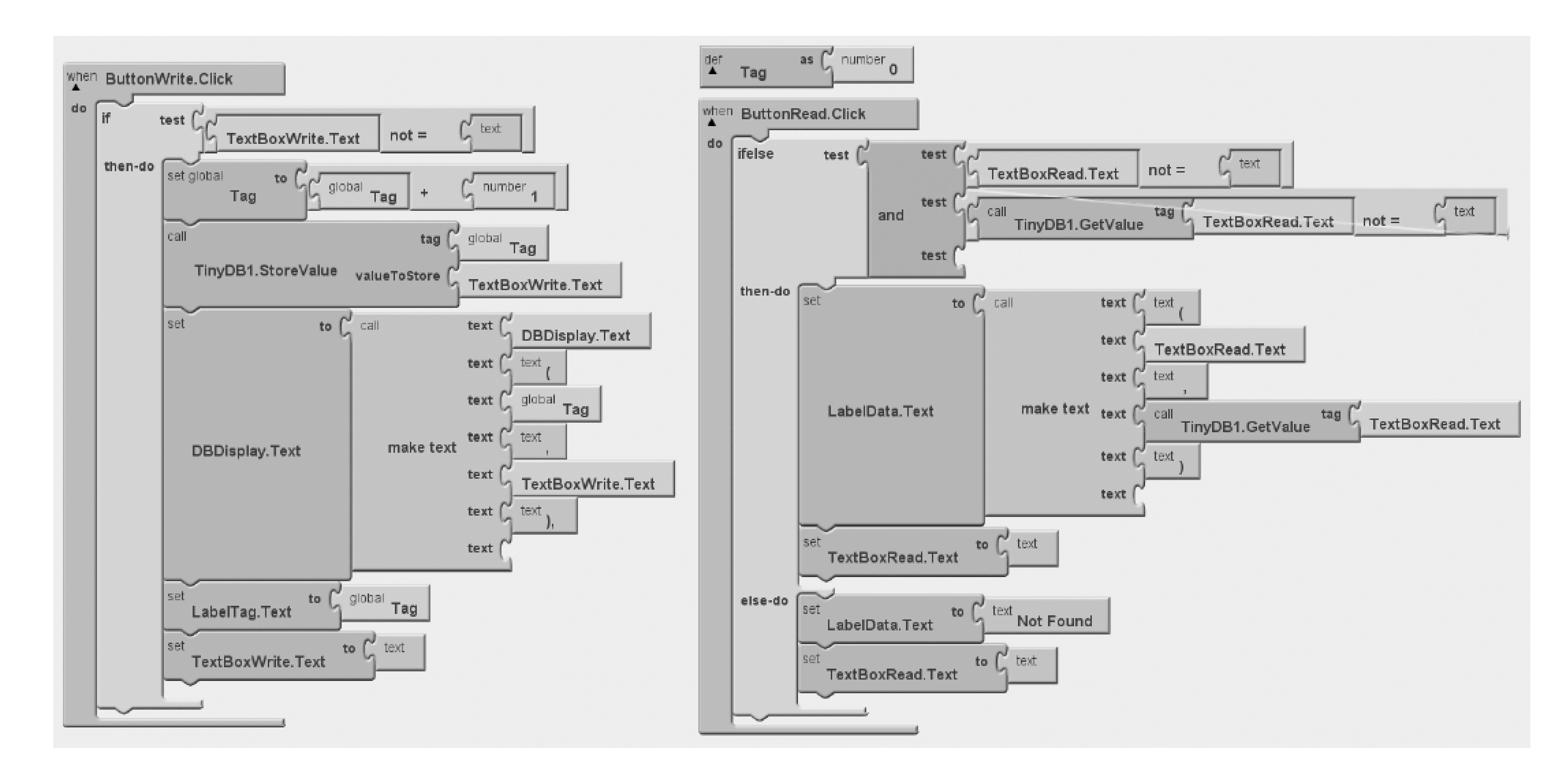


图 7.13 DataRecorder 示例的完整逻辑模块图

7.2 高级功能

7.1 节中介绍了 TinyDB 控件的数据存储和数据读取功能，本节对数据的更新和删除方法进行说明，以及使用 TinyDB 控件的一些注意事项。

TinyDB 控件没有独立的更新和删除数据的方法。如果数据存储中使用的标签在数据库中已经存在，则这次数据存储也可视为“数据更新”，原有数据会被新数据所替换。删除数据也同样要使用数据存储方法 StoreValue，将参数 valueToStore 赋上空值，如图 7.14 所示，原有数据将会被删除。

在一个应用程序中，多个 TinyDB 控件是共享存储空间的。即使使用多个 TinyDB 控件，标签仍然不能重复，否则会出现数据覆盖的情况，因此不建议在应用程序中使用多个 TinyDB 控件。例如在图 7.15 中，分别向数据库 TinyDB1 和 TinyDB2 存储一条数据，标签都为“0001”，先向 TinyDB1 存储数据“Tom”，再向 TinyDB2 存储数据“Lily”，此时 TinyDB1 和 TinyDB2 中标签“0001”的数据都是“Lily”。但需要注意的是，不同应用程序的数据库是各自独立的，不能通过 TinyDB 控件在不同应用程序之间传送数据。



图 7.14 通过写入空值来删除数据



图 7.15 TinyDB 控件共享存储空间示例

在 7.1 节的示例中，使用自动递增的数字(变量标签)作为数据存储的标签，这种方式简便、高效，而且不会出现标签重复的情况。变量标签适合有规律、数据量大的情况，对于突发性、数据量少的情况并不适合。

对于少量的数据，开发人员一般会采用自定义标签。使用自定义标签的好处显而易见，首先自定义标签比变量标签更为灵活，而且便于搜索数据；其次，自定义标签在应用的开发过程中减少了开发工作量，简化了逻辑结构；最后，自定义标签也更适合一组多项的数据存储，如存储学生信息时，可能需要同时存储学生的姓名、学号、班级和专业等多项不同的信息，在这种情况下自定义标签就可以将同一组的条目关联起来。

当然，自定义标签也有受到限制的方面，最大的限制就是标签可能会出现重叠。例如，在录入学生信息的时候，重名的情况就可能导致数据库的数据出现覆盖或者误删除的情况。这个问题一方面可以通过在逻辑中添加限定来解决，如在存入数据之前先判断目标标签下的内容是否为空，如果不为空则停止数据存储并给出提示；另一种方法是在数据设计时，选择不会或不易出现重复的项目作为标签，例如学生的“学号”就比使用“姓名”作为标签更为恰当，因为学号和学生的条目一一对应，而且不会出现重复。

TinyDB 控件无法对数据库中的内容进行检索。仍然以学生档案的例子来说，当以

学号作为标签存储时，可以通过学号获取学生信息，但是不能以姓名或者班级等内容进行检索。

TinyDB控件中的数据只有当用户下载并安装应用程序之后才能有效。对于开发测试人员来讲，如果在测试过程中重新启动App Inventor应用程序或者中断调试链接，数据库的内容也被清空，因为这一过程等同于将应用从手机当中移除之后重新载入。

7.3 示例——注册登录

在Register示例中，使用TinyDB控件建立一个数据库，用于保存用户的用户名、密码和公司名的信息，示例有“新用户注册”功能和“用户登录”功能，如图7.16所示。用户在“新用户注册”时填写的信息，会记录在数据库中，在“用户登录”时进行验证。

在用户成功登录后会进入用户信息界面，可以查看用户和公司名称信息，并可以进行删除用户操作，如图7.17所示。

Register
用户信息
注册新用户
用户登录
用户名：
密码：
公司名：
用户名：
密码：
登录
注册
Register new user or log in

图7.16 Register示例的注册登录界面



图7.17 Register示例的用户信息界面

垂直布局ModuleDisplay是用来承载“用户信息界面”的布局，该布局在程序启动时是被自动隐藏的，只有当用户登录成功后才显示。但为了便于界面的设计与开发，在界面编辑器中将ModuleDisplay的Visible的属性设置为showing，在界面开发完成后，运行应用程序前，将Visible的属性设置为hiding。

Register示例的界面示意图如图7.18所示。

Register示例的逻辑部分主要是处理4个按钮的单击事件，分别是Register按钮、Login按钮、Confirm按钮和Delete按钮的单击事件。

在用户单击Register按钮进行注册时，用户注册首先要判断用户所填写的信息是否完整，然后判断用户提供的用户名在数据库中是否已经存在，如果用户信息完整且用户名在数据库中不存在，则将用户提供的用户名、密码和公司名存储到数据库中，最后清空用户在界面上填写的数据。在数据存储过程中，每次成功注册都产生两条数据：(用户名，密码)和(用户名+Corporation，公司名)，其中“用户名”和“用户名+Corporation”是这两条数据的标签。Register按钮单击事件模块如图7.19所示。



图 7.18 Register 示例的界面示意图

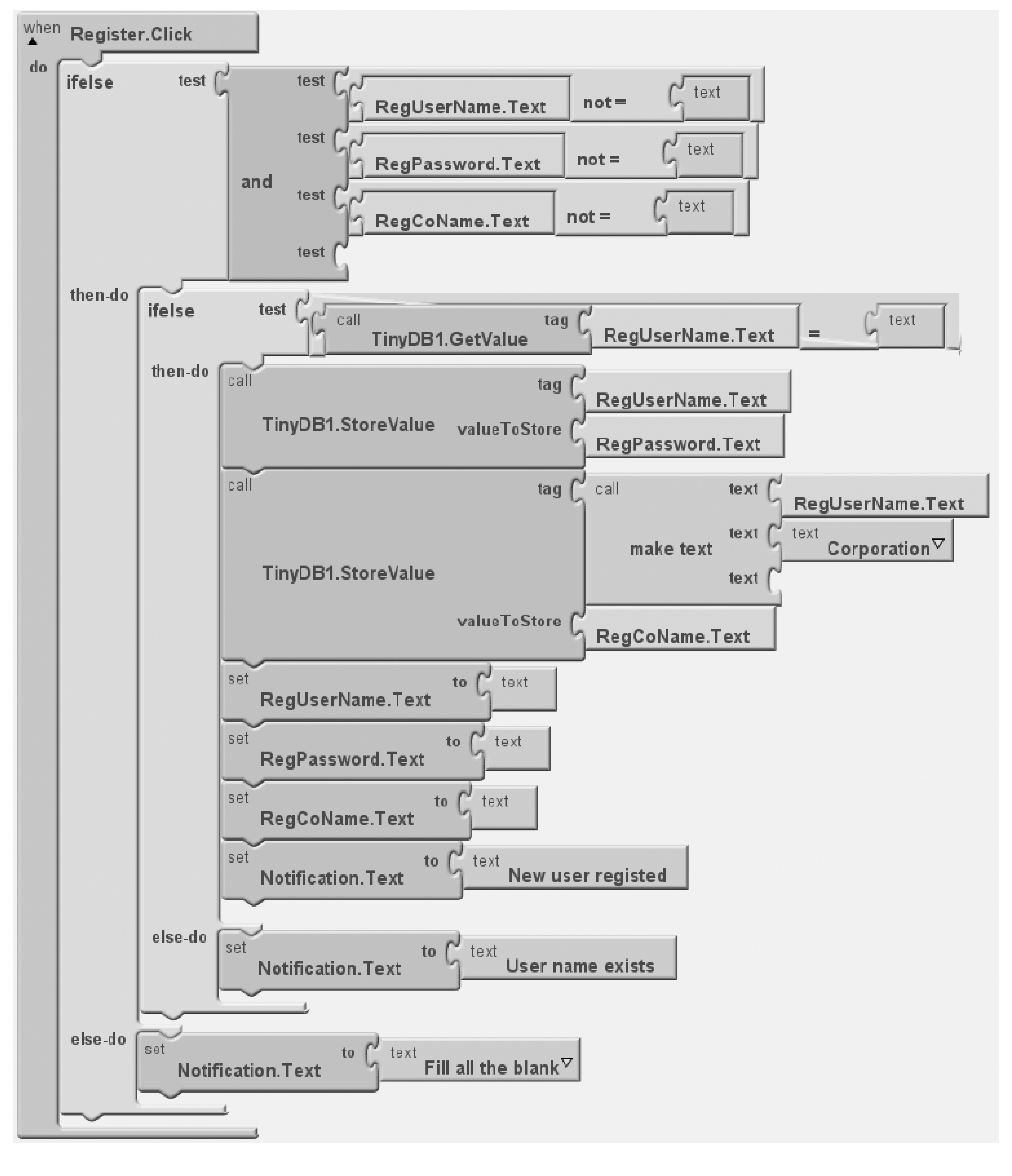


图 7.19 Register 按钮单击事件

在用户单击 Login 按钮进行登录时，首先要判断用户所填写的用户名和密码信息是否完整，然后判断用户名在数据库中是否存在，如果用户提供完整的用户名和密码，且用户名在数据库中存在，则将用户名作为标签在数据库中获取"原始密码"，并将用户提供的密码与原始密码进行比较，如果密码匹配则用户登录成功，否则登录失败。登录成功后，显示"Logged in"的提示信息，并将登录和注册的界面布局 ModuleRegAndLogin 的 Visible 属性设置为 false，隐藏登录和注册的界面；并将用户信息的界面布局 ModuleDisplay 的 Visible 属性设置为 true，显示用户信息界面。Login 按钮单击事件模块如图 7.20 所示。

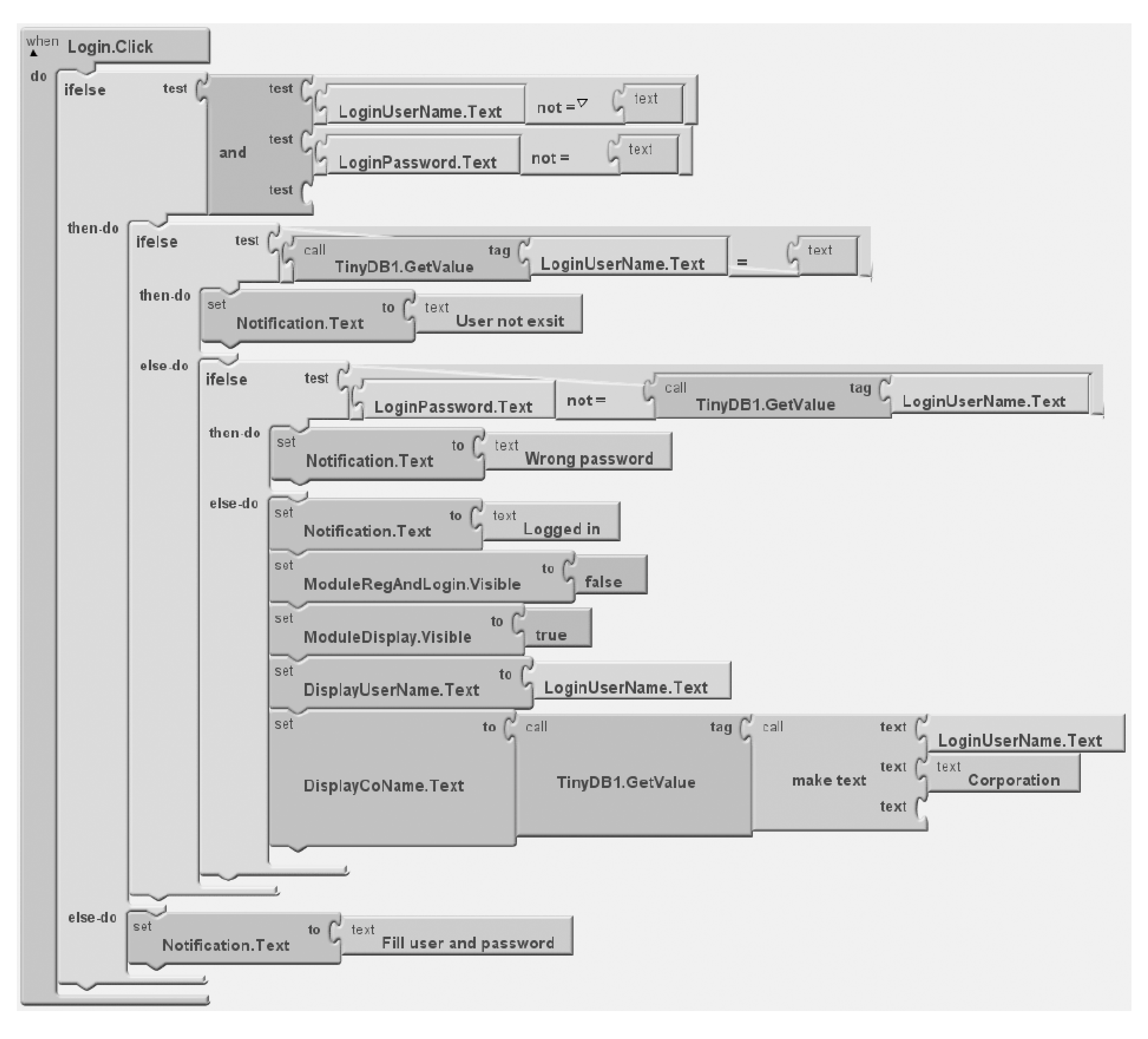


图 7.20　Login 按钮单击事件

在进入用户信息界面后，如果用户单击 Confirm 按钮，会显示提示信息"Regist new user or log in"，并将登录和注册的界面布局 ModuleRegAndLogin 的 Visible 属性设置为 true，重新显示登录和注册的界面；同时将用户信息的界面布局 ModuleDisplay 的 Visible 属性设置为 false，隐藏显示用户信息界面。Confirm 按钮单击事件模块如图 7.21 所示。

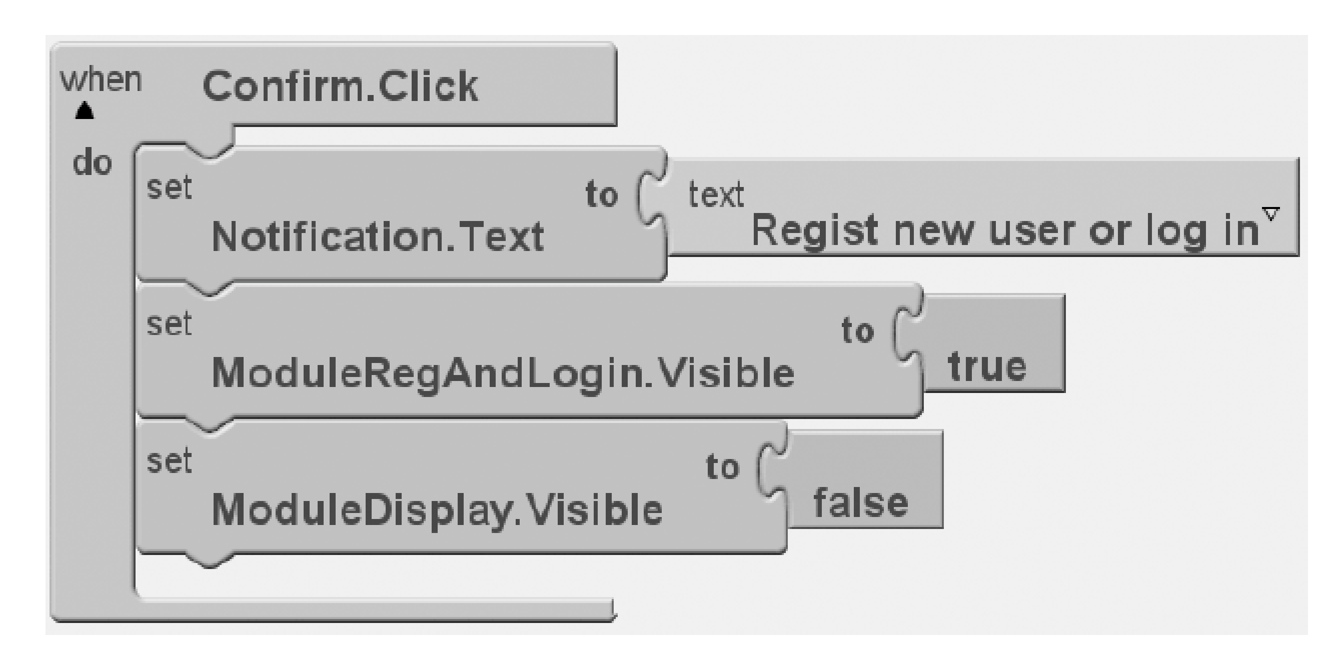


图 7.21　Confirm 按钮单击事件

在进入用户信息界面后，如果用户单击 Delete 按钮，会显示提示信息“User deleted”，同时显示登录和注册界面，隐藏显示用户信息界面，并将数据库中的用户信息删除。Delete 按钮单击事件模块如图 7.22 所示。

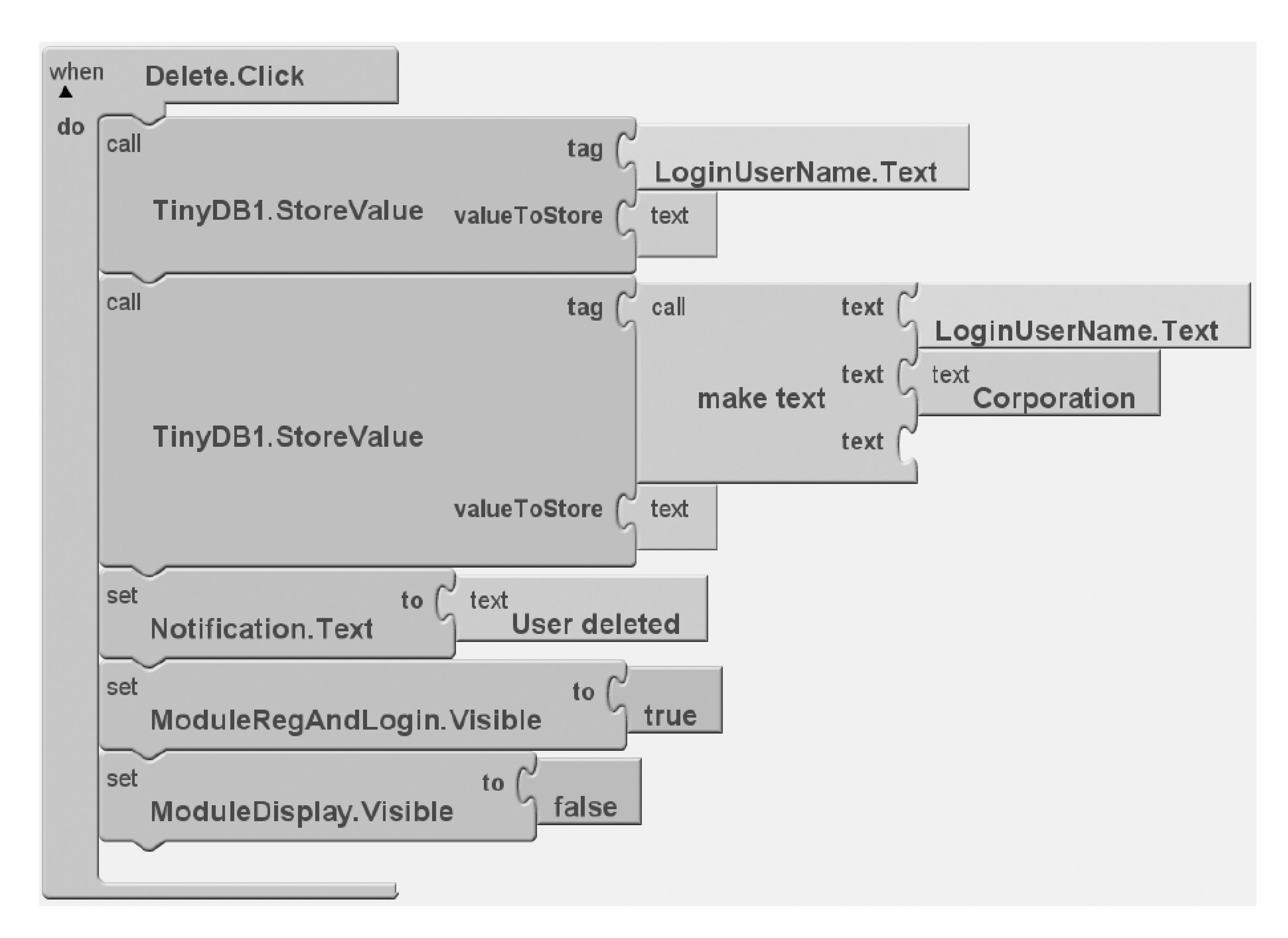


图 7.22　Delete 按钮单击事件

Register 示例的全部逻辑模块如图 7.23 所示。

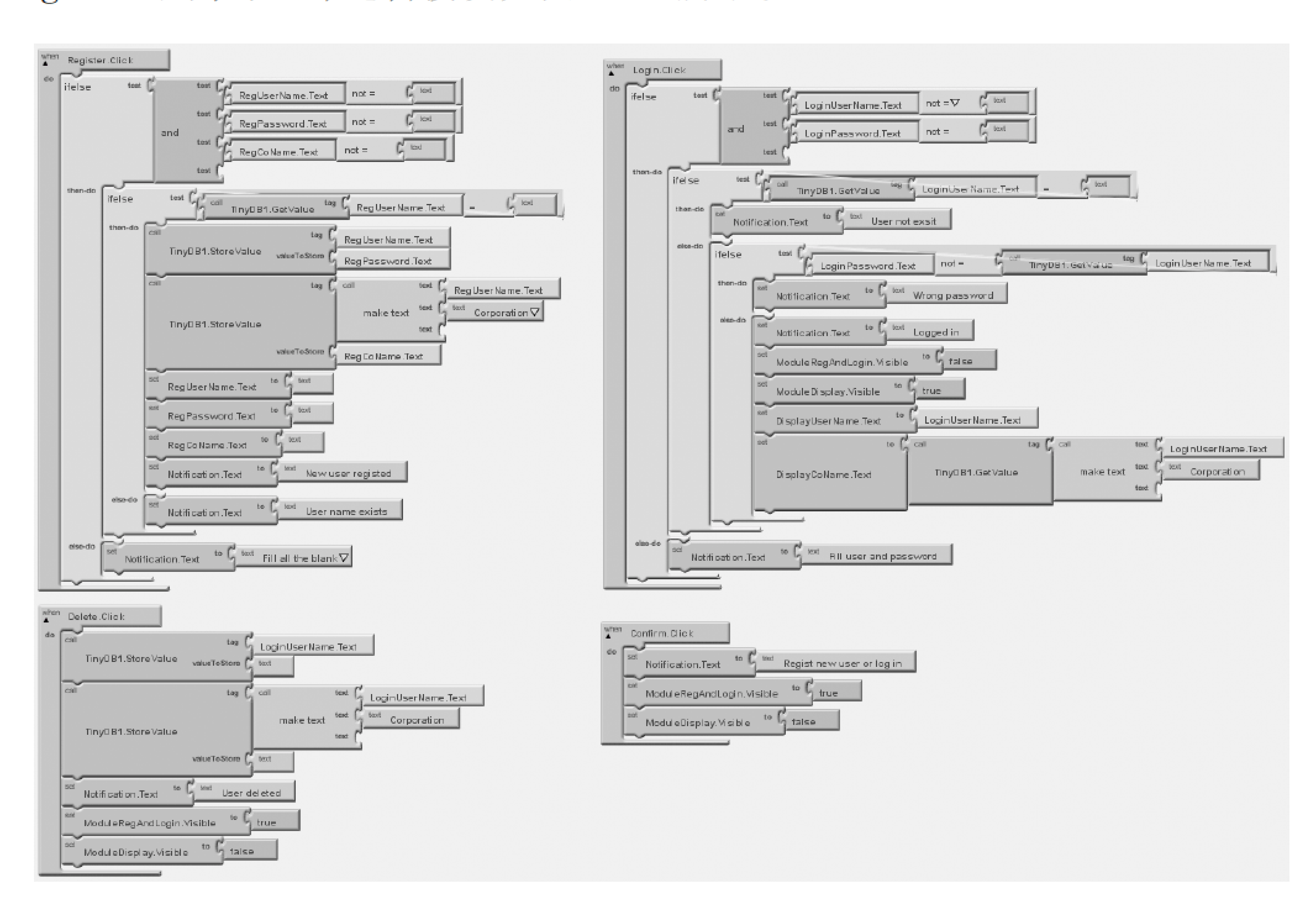


图 7.23　Register 示例的全部逻辑模块

Register 示例实现了用户数据的长期保存，在应用程序重新启动之后依然有效，仍然可以通过注册成功的用户名和密码进行登录。

习 题

1. 简述 TinyDB 控件实现基本的数据存储与访问功能的逻辑步骤。

2. 如果在一个应用程序中使用两个 TinyDB 控件，能够实现两个独立的数据库吗？为什么？

3. 如何在 TinyDB 控件中实现表项的删除和更新操作？

4. 编程实现“抽签”程序。分为写签和抽签两部分，在写签部分用户可以在 1～10 的序号中设置一个中奖签，在抽签部分用户随机在 1～10 序号中抽签，并显示抽签结果。

第8章



地图应用开发

地图开发是当前最流行的应用,也是最有潜在需求的领域。通过本章的学习,读者可以掌握位置传感器的使用方法,以及如何在应用程序上使用谷歌地图。

学习目标:

- 掌握位置传感器的使用方法;
- 了解不同位置信息获取途径;
- 掌握通知控件的使用方法;
- 掌握谷歌地图的使用方法。

8.1 位置传感器

位置服务(Location-Based Services,LBS),又称定位服务或基于位置的服务,融合了GPS定位、移动通信、导航等多种技术,提供与空间位置相关的综合应用服务。定位服务可以获取用户终端的位置信息,Android系统支持GPS、WiFi和基站信号三种定位方式。

位置传感器(LocationSensor)采用上述三种定位技术,可以获取手机的经度、纬度和海拔等数据。位置传感器是非可视化控件,在界面编辑器中的显示内容如图8.1所示。

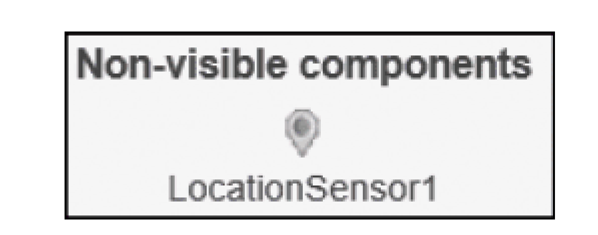


图8.1　位置传感器

位置传感器支持较多的属性,包括定位精度、定位硬件、经纬度、海拔等信息,全部属性信息和说明如表8.1所示。

表8.1　位置传感器属性

属　　性	属性说明
Accuracy	设备的精确度,单位:米
AvailableProviders	可用的位置服务提供硬件
CurrentAddress	当前所在位置地址
Enabled	是否启用位置服务
HasAccuracy	是否可返回设备精确度
HasAltitude	是否可返回设备高度

续表

属　　性	属 性 说 明
HasLongitudeLatitude	是否可返回设备经纬度
Latitude	纬度
Longitude	经度
Altitude	海拔高度
ProviderLocked	锁定位置服务提供者
ProviderName	位置服务提供者名称
TimeInterval	每隔多长时间显示一次定位信息
DistanceInterval	每隔多大距离显示一次定位信息

位置传感器支持位置改变事件(LocationChanged)和位置服务提供者状态改变事件(StatusChanged),如图8.2所示。位置改变事件在手机的经度、纬度和高度发生变化时产生,一般用来获取这三项数值。位置服务提供者状态改变事件在位置服务提供者的状态发生变化时产生,用来获取位置服务提供者的基本信息和状态信息。

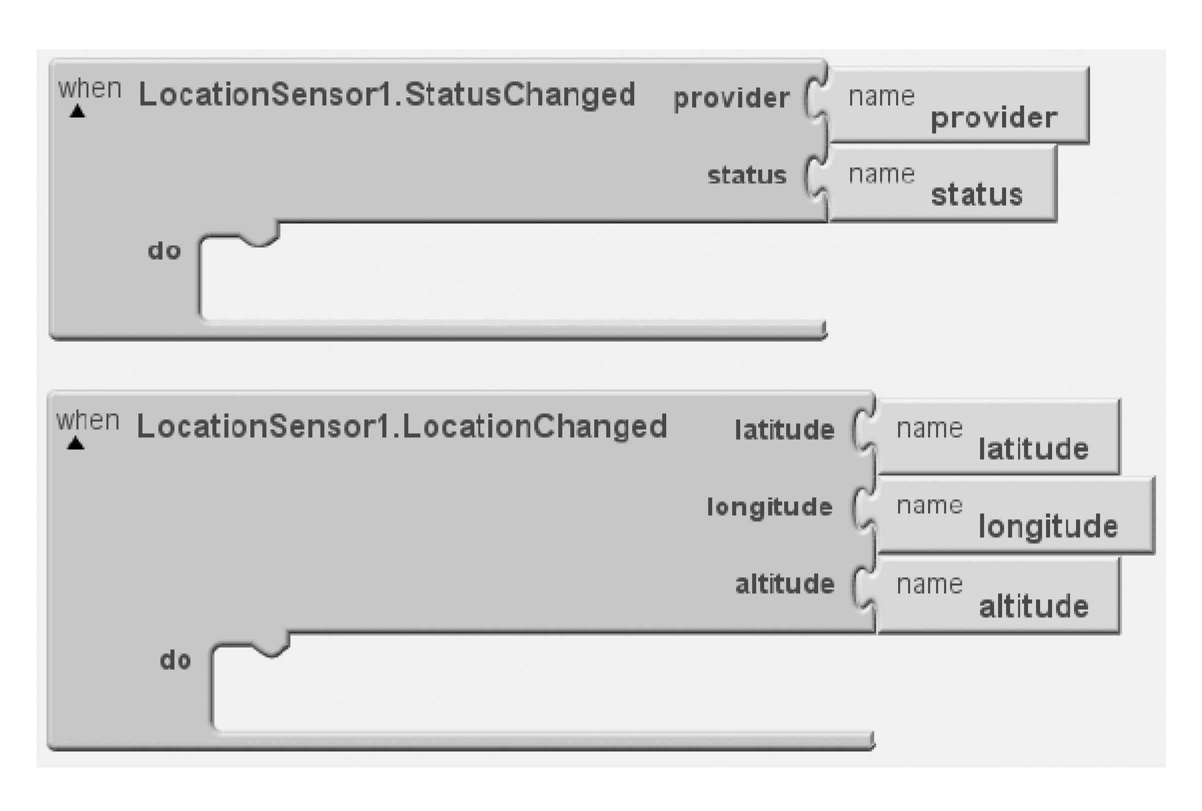


图8.2　位置传感器支持的事件

位置传感器支持LatitudeFromAddress方法和LongitudeFromAddress方法,如图8.3所示。LatitudeFromAddress方法可以从地址中获取经度信息,LongitudeFromAddress方法可以从地址中获取纬度信息。

为了能更好地理解位置传感器的事件和属性,下面介绍可以获取位置信息和位置服务提供者信息的LocationSensor示例。LocationSensor示例的运行界面如图8.4所示。



图8.3　位置传感器支持的方法



图8.4　LocationSensor示例的运行界面

在手机上运行该示例，可以获取到手机的经度和纬度信息，笔者的海拔信息、服务提供者信息和状态信息处于未知状态。

图 8.5 是 LocationSensor 示例的界面示意图，该示例中可视化控件只有标签，非可视化控件有表格布局和位置传感器。

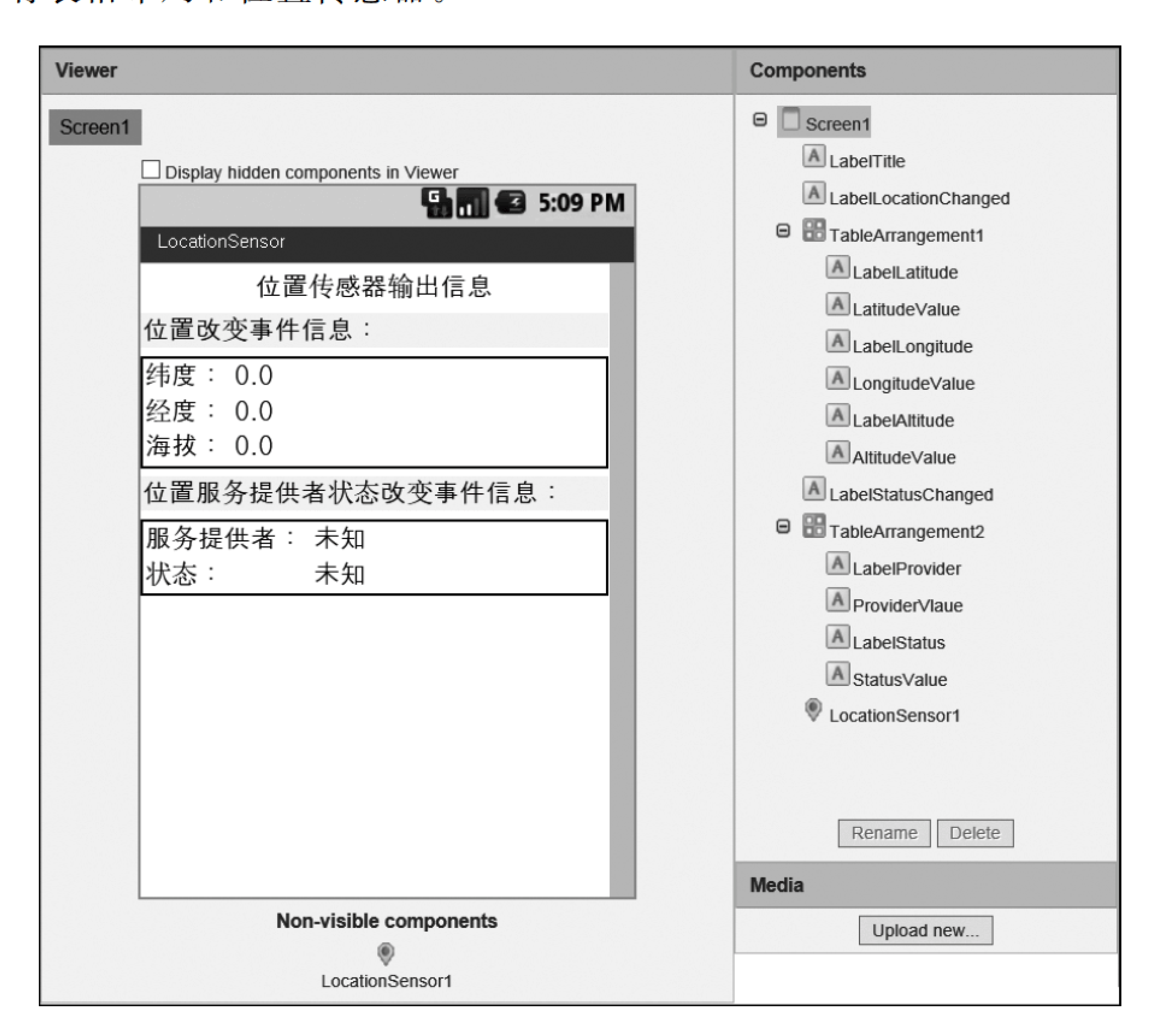


图 8.5 LocationSensor 示例界面示意图

LocationSensor 示例的逻辑模块比较简单，只有两个关于位置传感器控件的事件模块，LocationSensor 示例的全部逻辑模块如图 8.6 所示。

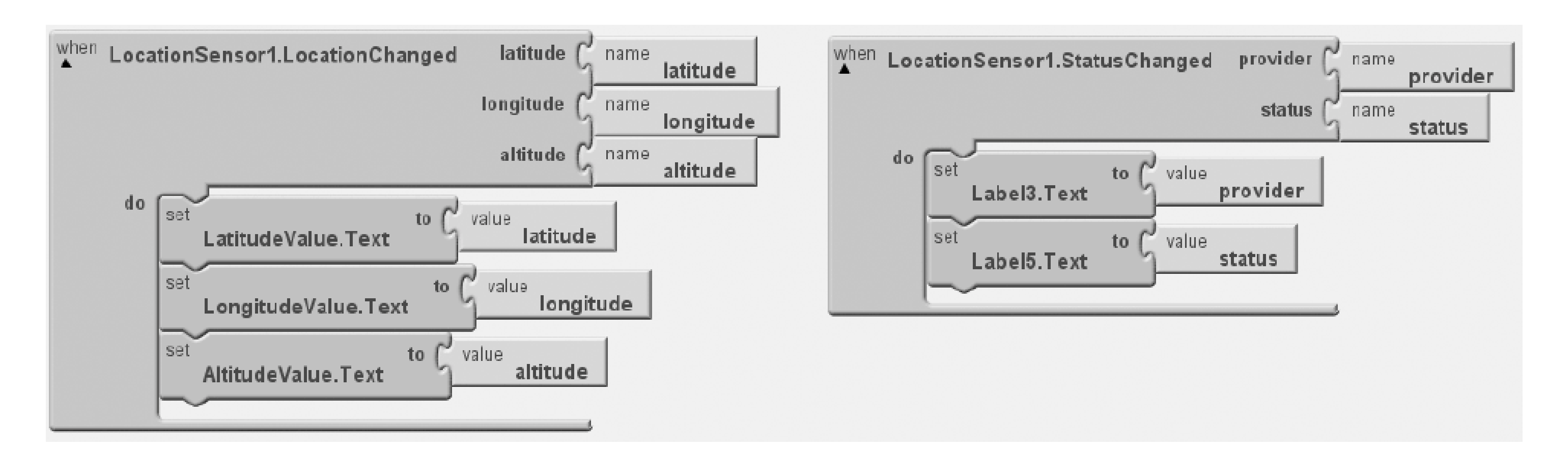


图 8.6 LocationSensor 示例的全部逻辑模块

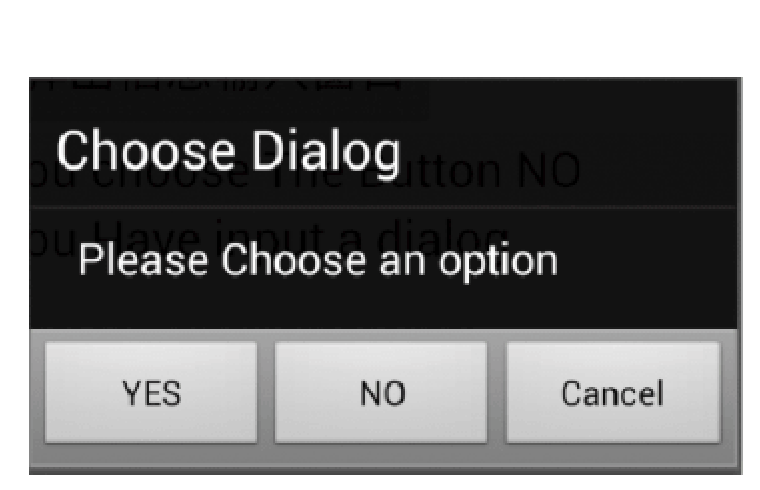
8.2 通知控件

在进入 8.3 节“谷歌地图”前，先介绍一个重要的通知控件 Notifier，这是一个非可视化控件，如图 8.7 所示。

通知控件提供了多种不同的方式与手机用户交互信息，例如在屏幕中出现的浮动消息，以及弹出的选择对话框或输入对话框，如图 8.8 所示。




图 8.7　通知控件



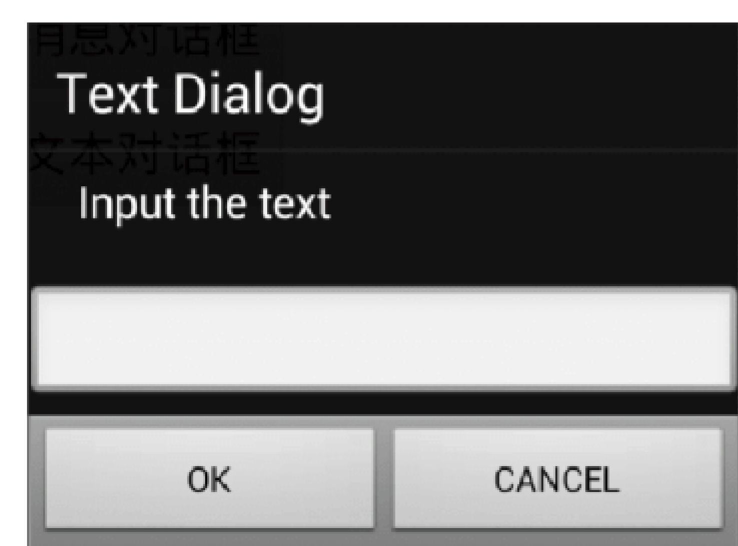


图 8.8　Notifier 的弹出窗口

Notifier 是一个没有属性的控件，支持选择后事件（AfterChoosing）和输入后事件（AfterTextInput），如图 8.9 所示。选择后事件，用户在选择对话框中做出选择后产生，一般与 ShowChooseMessageDialog 方法联合使用；输入后事件，用户在文本对话框中输入并返回后产生，一般与 ShowTextDialog 方法联合使用。

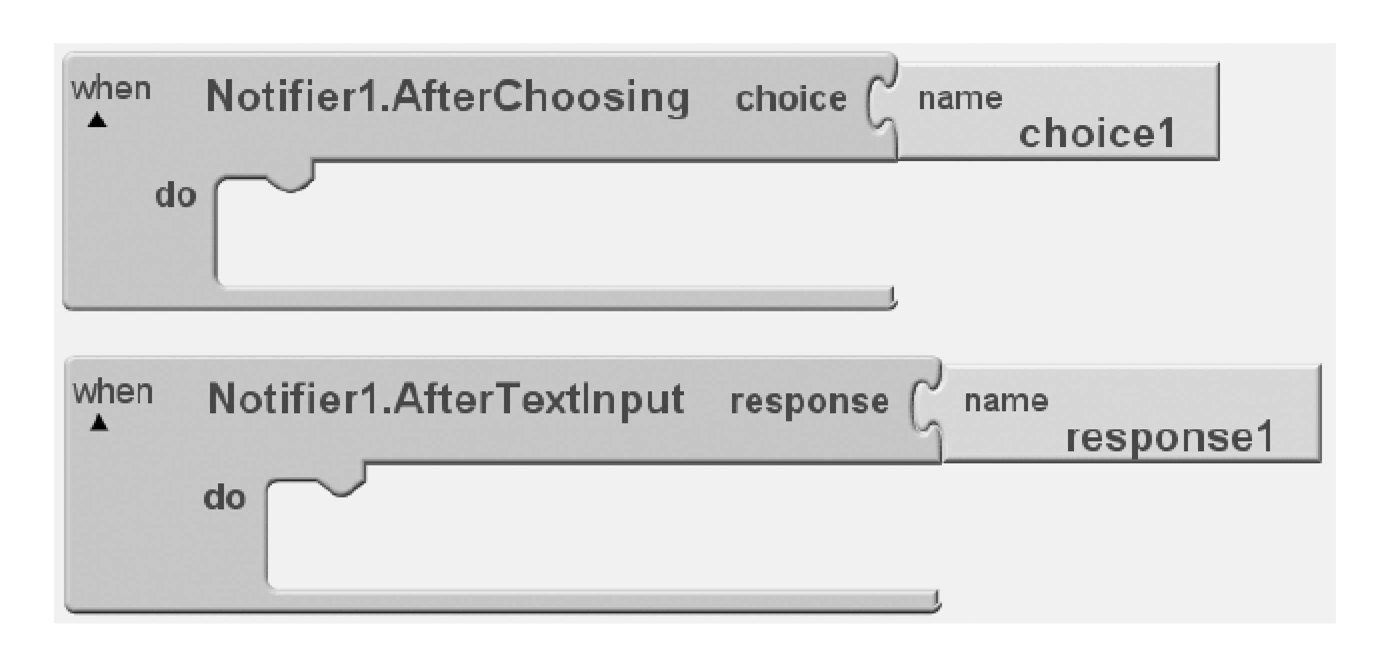


图 8.9　Notifier 事件

Notifier 支持 7 种方法，包括显示消息对话框、显示选择对话框和显示文本对话框等，如表 8.2 所示。

表 8.2　通知控件方法

方　　法	说　　明
ShowMessageDialog	显示消息对话框，只有一个按钮，可设定按钮显示的文字
ShowChooseDialog	显示选择对话框，有两个或三个按钮，并可设定按钮显示的文字
ShowTextDialog	显示文本对话框，可在对话框中输入文字
ShowAlert	显示警告信息
LogError	错误信息
LogInfo	提示信息
LogWarning	警告信息

下面用 Notifier 示例说明如何使用通知控件产生消息对话框，获取用户在选择对话框中的选择和在文本对话框中的输入。Notifier 示例的运行界面如图 8.10 所示，用户在

单击不同的按钮后，会产生一个与按钮内容相对应的对话框或者提示信息。

Notifier 示例的界面示意图如图 8.11 所示，界面主要由按钮和标签组成，只有一个非可视化控件 Notifier1。

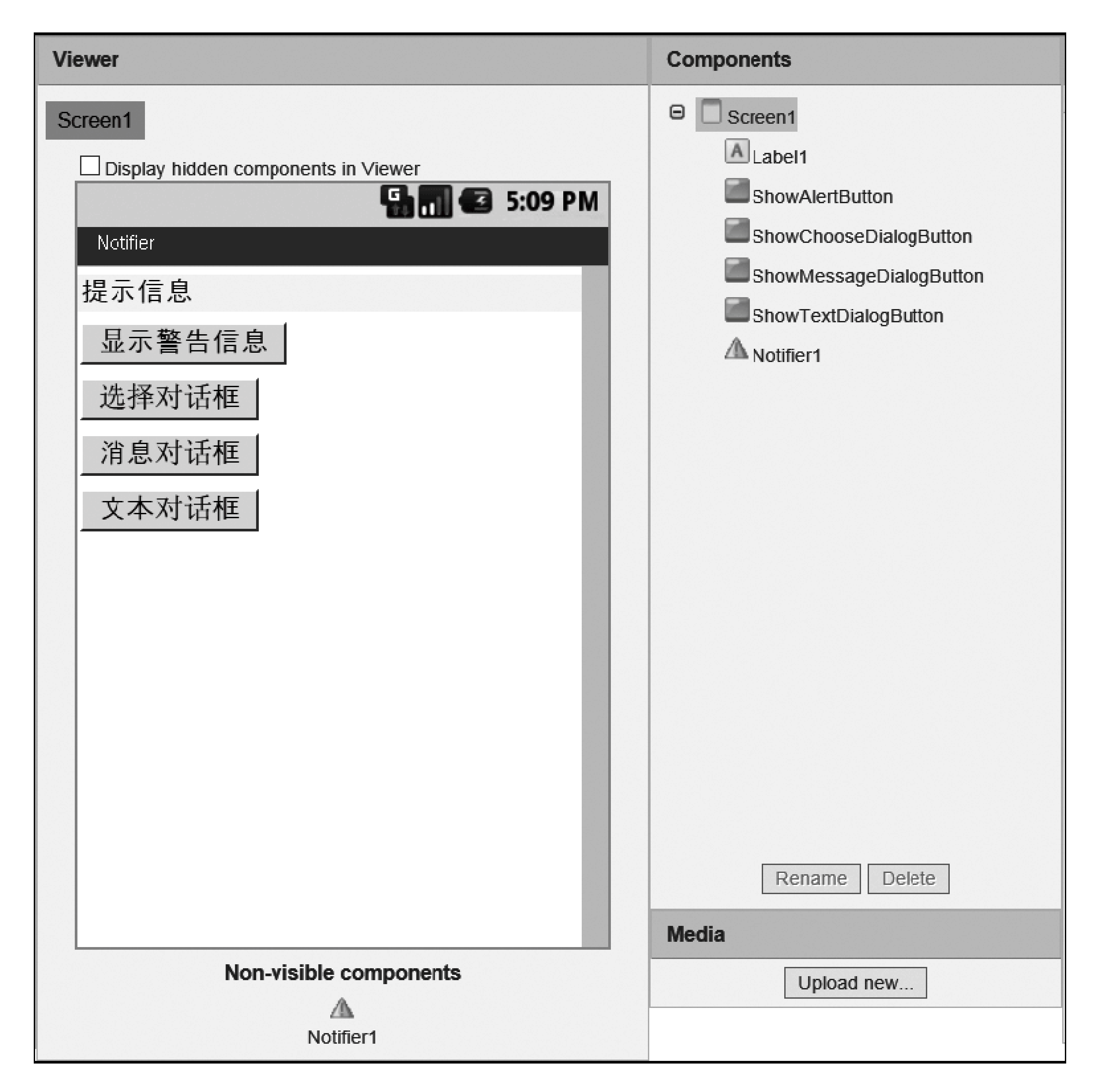


图 8.10 Notifier 示例　　　　图 8.11 Notifier 示例界面示意图

在 ShowAlertButton 按钮的单击事件中，调用通知控件的 ShowAlert 方法，在手机屏幕上显示浮动的提示信息，如图 8.12 所示。

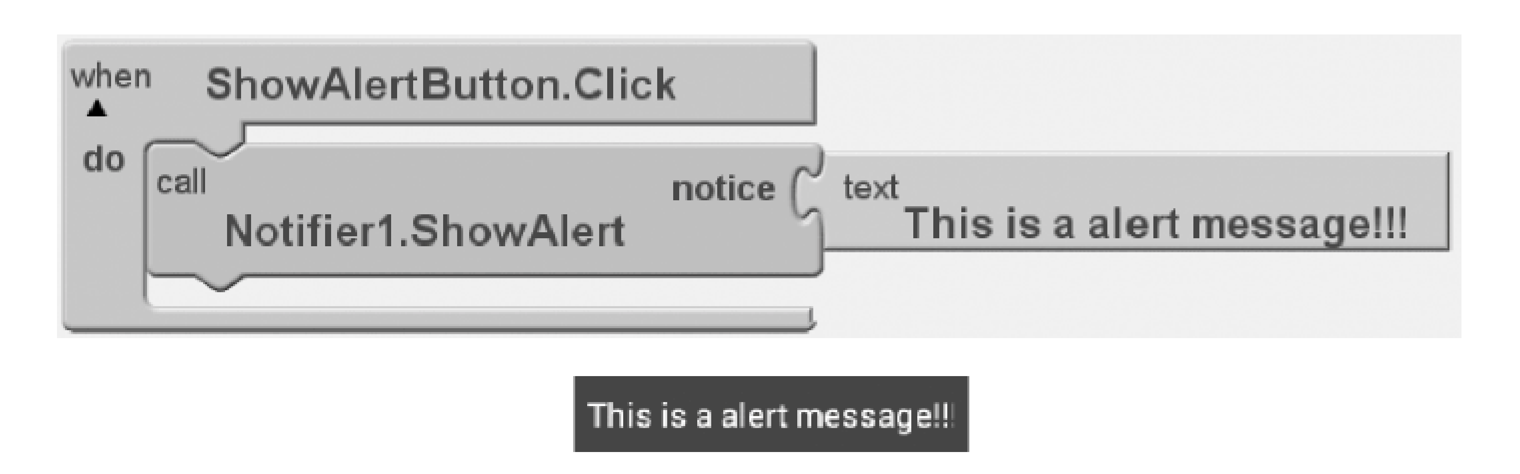
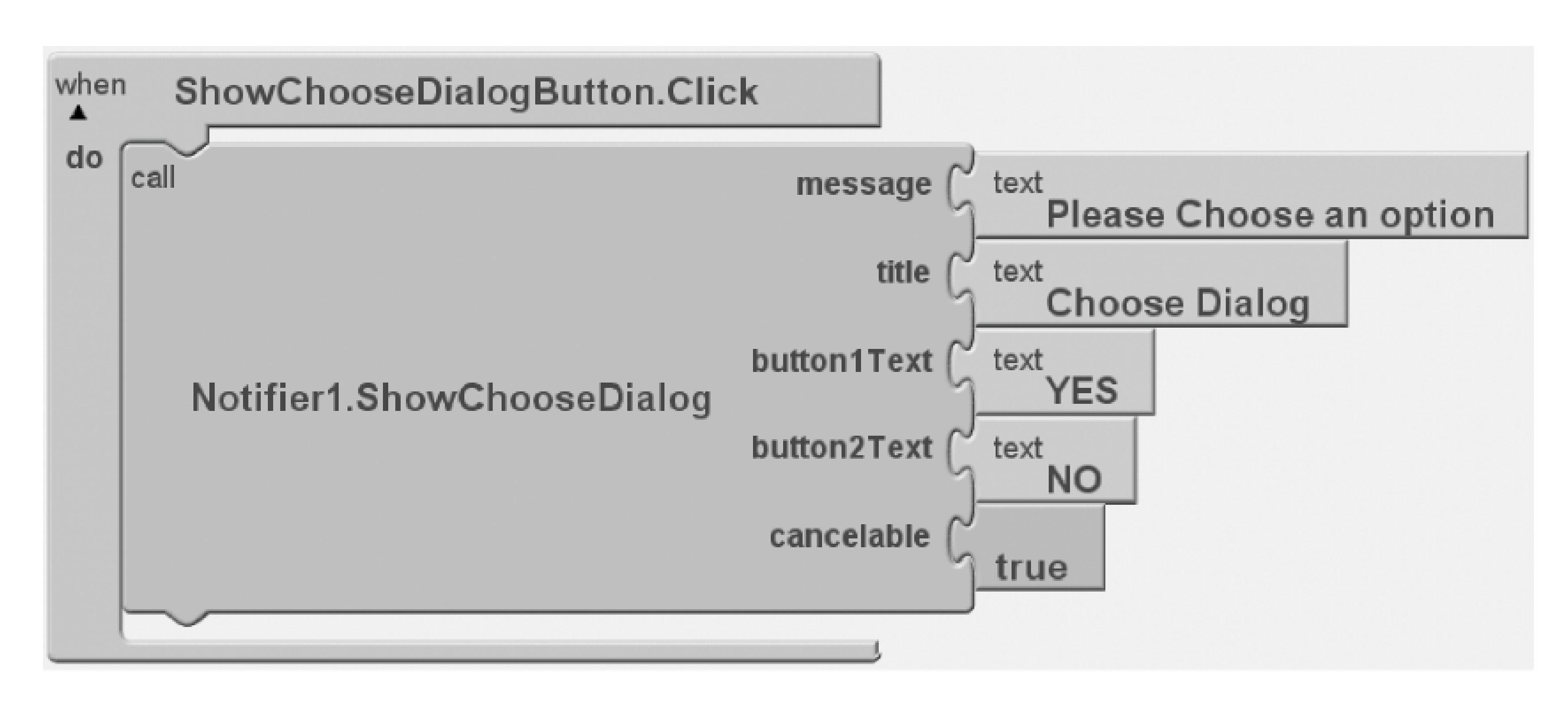


图 8.12 按钮单击事件和警告信息

在 ShowChooseDialogButton 按钮的单击事件中，调用通知控件的 ShowChooseDialog 方法，在手机屏幕上显示选择对话框，如图 8.13 所示。模块各个参数与选择对话框中显示内容的对应关系，读者很容易从图中找到。参数 cancelable 控制着 Cancel 按钮，如果设置为 true，则在选择对话框中出现 Cancel 按钮，否则将不出现 Cancel 按钮。

当用户在选择对话框中做出选择后，将产生 AfterChoosing 事件，参数 choice 代表用户的选择，用按钮上的文字表示。例如，如果用户选择的是 YES 按钮，则参数 choice 是字符串“YES”；如果用户选择的是 Cancel 按钮，则参数 choice 是字符串“Cancel”。在 AfterChoosing 事件中，只是将用户的选择显示在标签 Label1 中，如图 8.14 所示。



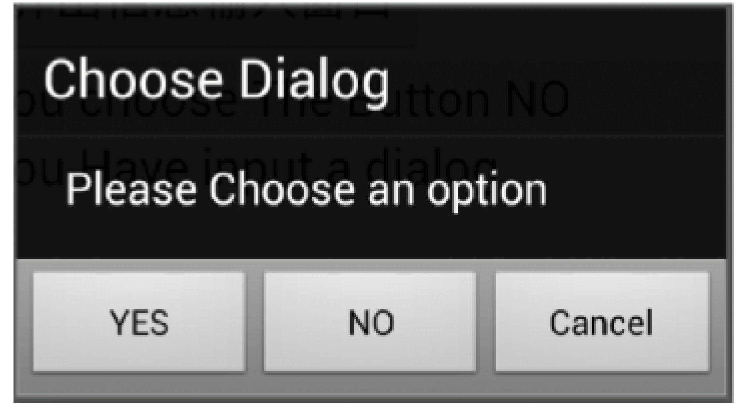


图 8.13　按钮单击事件和选择对话框

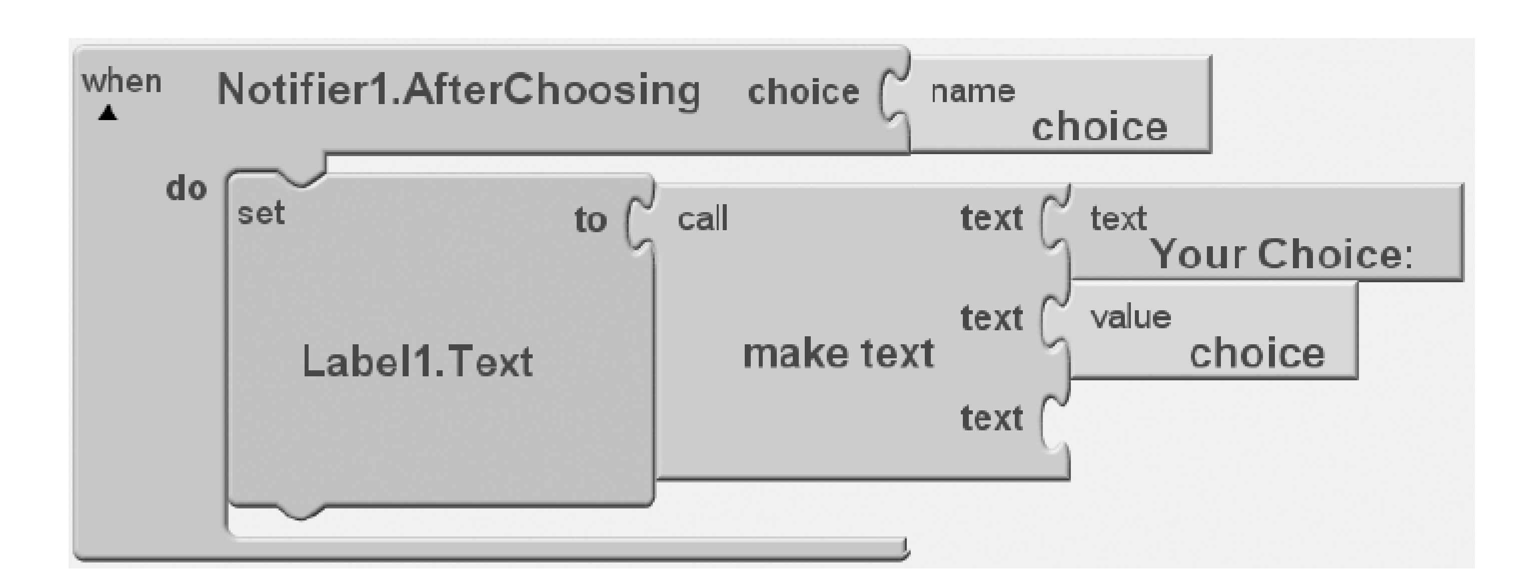
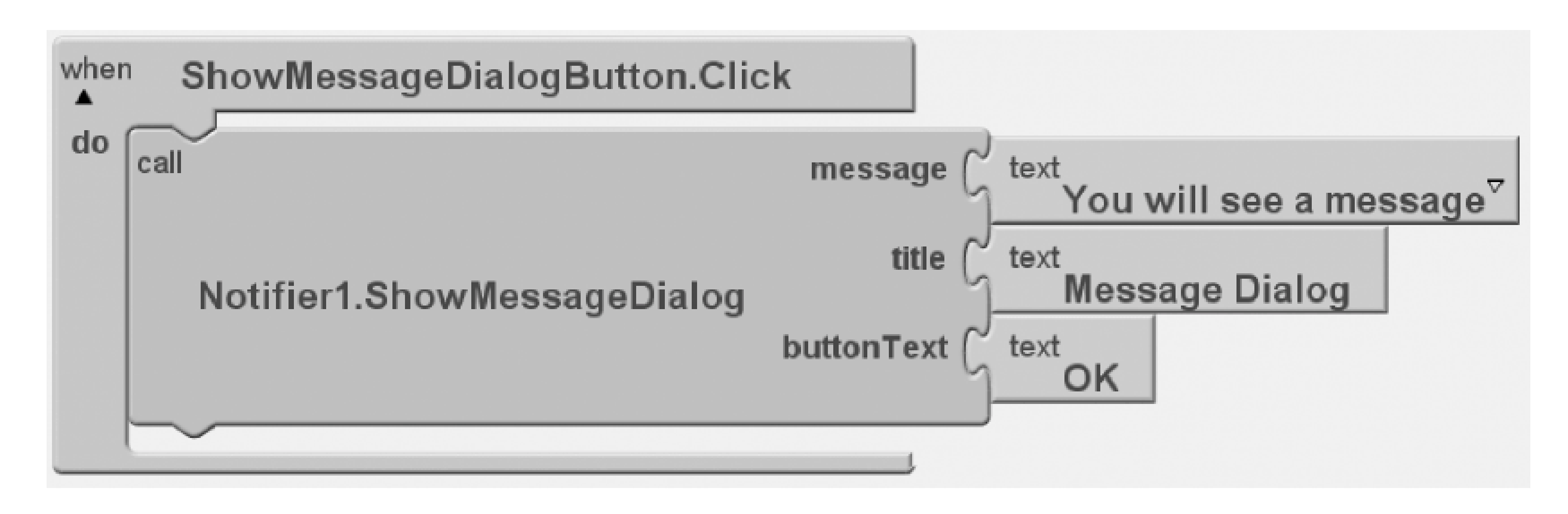


图 8.14　AfterChoosing 事件

在 ShowMessageDialogButton 按钮的单击事件中，使用通知控件的 ShowMessageDialog 方法，显示消息对话框，如图 8.15 所示。参数 message 是消息对话框中的信息；参数 title 是消息对话框的标题；参数 buttonText 是消息对话框中唯一一个按钮的文字提示内容。



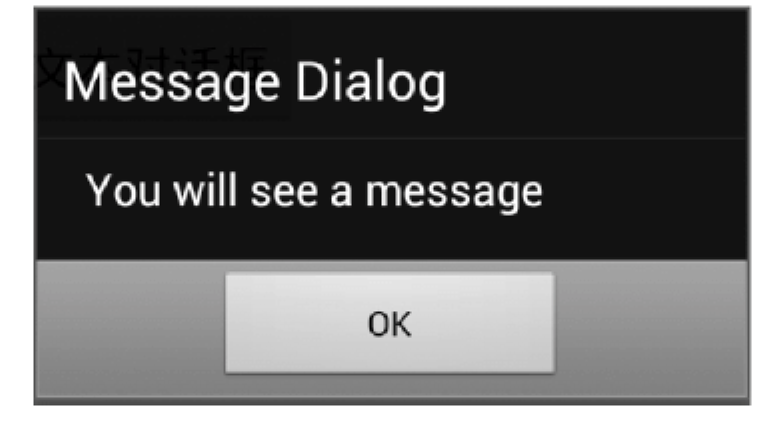


图 8.15　按钮单击事件和消息对话框

在 ShowTextDialogButton 按钮的单击事件中，使用通知控件的 ShowTextDialog 方法，显示文本对话框，如图 8.16 所示。



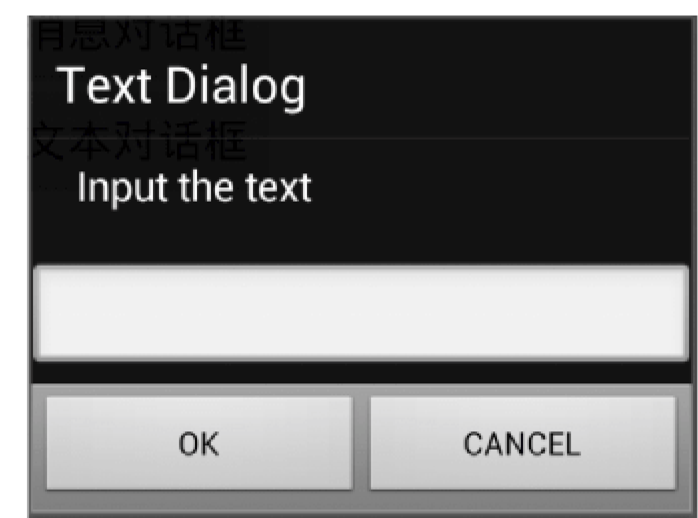


图 8.16　按钮单击事件和文本对话框

用户在文本对话框中输入信息后，如果单击 OK 按钮，则会触发 AfterTextInput 事件，输入的信息会被传递到参数 response 中；但如果用户单击的是 CANCEL 按钮，参数 response 将获取到字符串“CANCEL”。在 AfterTextInput 事件中，只是将用户输入的信息显示在标签 Label1 中，如图 8.17 所示。

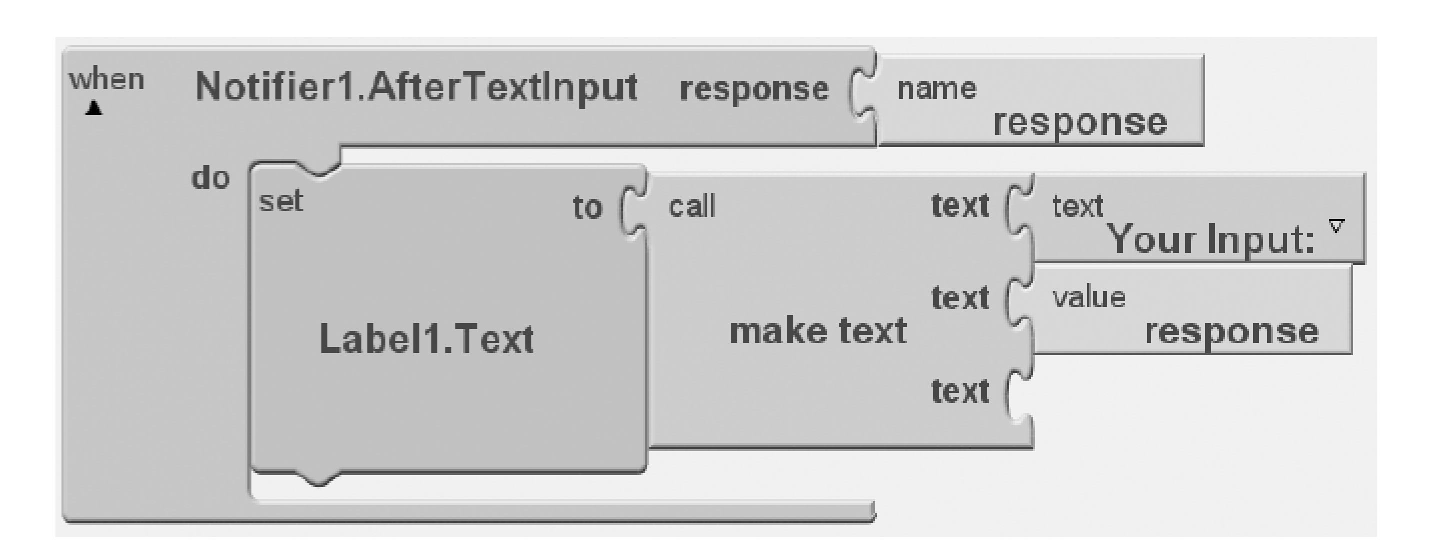


图 8.17　AfterTextInput 事件

Notifier 示例的全部逻辑模块如图 8.18 所示。

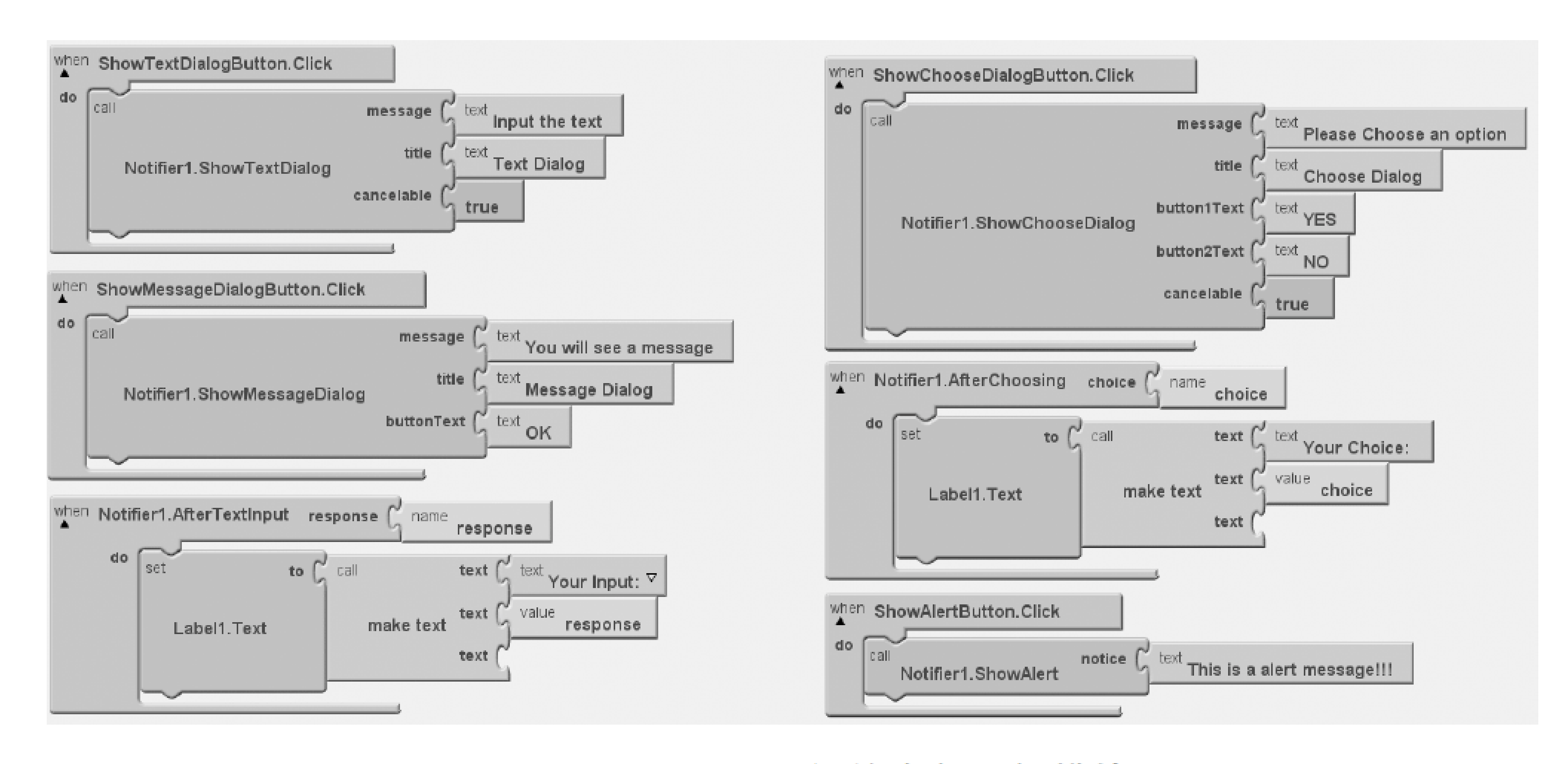


图 8.18　Notifier 示例的全部逻辑模块

8.3 谷歌地图

谷歌地图是谷歌公司提供的电子地图服务，可以提供含有政区、交通和商业信息的地图，提供不同分辨率的卫星照片，并且可显示地形和等高线地形视图。图 8.19 是 Android 手机中的谷歌地图。



图 8.19　Android 手机的谷歌地图

在 App Inventor 中使用谷歌地图一般有两种途径，一种是使用 WebViewer（在 Not ready for prime time 中），另一种是使用 ActivityStarter（在 Other stuff 中）。

使用 WebViewer 是在浏览器中打开谷歌地图，只需将 URL 链接地址传递给 WebViewer 控件的 GoToUrl 方法就可以在 Web 浏览器中打开谷歌地图。例如，希望获取经纬度为(45.76，126.70)的谷歌地图，GoToUrl 方法的参数构成为“http：//maps.google.com/maps?q=45.76，126.70”，显示效果如图 8.20 所示。不推荐使用这种方法，因为使用的并不是正式控件，而且显示速度和效果并不十分理想。



图 8.20　手机浏览器的谷歌地图

使用 ActivityStarter 是在新的屏幕页中打开谷歌地图，只需将 URI 参数传递给 ActivityStarter 控件的 DataUri 方法，就可以在新的屏幕页中打开谷歌地图。这种方法的地图显示速度和效果较好。下面将详细介绍 ActivityStarter 控件的属性、事件和方法，然后通过一个示例介绍如何使用 ActivityStarter 浏览谷歌地图。

ActivityStarter 是启动其他程序屏幕页的控件，可以启动其他 App Inventor 程序、摄像头程序、Web 搜索程序和谷歌地图。ActivityStarter 是非可视化控件，在界面控制器中的显示如图 8.21 所示。

图 8.21　ActivityStarter 控件

ActivityStarter 控件只有一个 AfterActivity 事件，在打开屏幕页(Activity)后产生该事件，其参数 result 常用于获取屏幕页的返回值，如图 8.22 所示。

ActivityStarter 有两个方法：StartActivity 和 ResolveActivity。StartActivity 方法用来启动目标屏幕页，在相关参数设置完毕后，调用该方法可以在手机上直接打开新的应用程序。ResolveActivity 用来获取屏幕页的解析结果，如果没有明确指定需要打开哪个应用程序，则需要由 Android 系统来确定，这个方法可以在正式打开程序前，尝试获取 Android 系统的解析结果。如果 ResolveActivity 方法的解析结果为空，则表示没有可以被打开的应用程序。ActivityStarter 方法如图 8.23 所示。



图 8.22　AfterActivity 事件



图 8.23　ActivityStarter 方法

通过设置 ActivityStarter 的属性可以指定启动的程序，每项属性的含义可以参考表 8.3。

表 8.3　ActivityStarter 属性

属　　性	说　　明	属　　性	说　　明
Action	动作	ExtraValue	键值
ActivityClass	类名	Result	结果返回值
ActivityPackage	包名称	ResultName	结果返回值名称
DataUri	通用资源符	ResultType	结果返回值类型
DataType	数据类型	ResultUri	返回值的通用资源标识或数据
ExtraKey	键名		

下面将要启动的其他应用程序分为 5 类，分别来介绍如何设置 ActivityStarter 的属性值来启动不同类型的应用程序。

1. 启动其他 App Inventor 应用程序

启动其他 App Inventor 应用程序只需要设置两个参数：ActivityPackage 和 ActivityClass。

首先要从 App Inventor 中下载目标程序的源代码，因为需要的两个参数值就在目标程序的源代码中。然后将源代码解压缩，并找到文件 youngandroidproject/project.properties，这个文件是保存应用程序基础数据的文件。

打开这个文件后，可以看到以“main”开头的第一行，这里就是需要找的内容。为了

说明如何获取 ActivityPackage 和 ActivityClass 参数，下面以 LocationSensor 示例 project. properties 文件内容进行说明。

```
main=appinventor.ai_wangxianghui2013.LocationSensorEvent.Screen1
name=LocationSensorEvent
assets=../assets
source=../src
build=../build
versioncode=1
versionname=1.0
```

文件的第一行是以"main"开始的，去掉"main="就是 ActivityClass 参数，也就是说 ActivityClass 参数的值应该是 appinventor. ai_wangxianghui2013. LocationSensorEvent. Screen1。将 ActivityClass 参数的最后一部分(. Screen1)去掉，就是需要的 ActivityPackage 参数，因此 ActivityPackage 参数为 appinventor. ai_wangxianghui2013. LocationSensorEvent。

2. 启动手机中已有的应用程序

启动手机中已有的应用程序需要设置三个参数：Action、ActivityPackage 和 ActivityClass。下面以摄像头程序为例，说明如何启动手机中已有的应用程序。

摄像头程序是手机中必备的内置软件，用来控制手机摄像头进行录像和拍照。如果要找到需要的参数，则需对基于代码的 Android 程序设计具有一定的了解，这方面的内容可以参考笔者的另一本 Android 书籍《Android 应用程序开发》。这里直接给出需要的参数，如表 8.4 所示。

表 8.4 启动手机程序的参数值

参　　数	值
Action	android. intent. action. MAIN
ActivityPackage	com. google. android. camera
ActivityClass	com. android. camera. Camera

3. 启动 Web 搜索程序

这里假设需要搜索的内容是"greatwall"，则需要设置的参数共 5 个，分别是 Action、ExtraKey、ExtraValue、ActivityPackage 和 ActivityClass，具体参数设置如表 8.5 所示。

表 8.5 启动 Web 搜索程序的参数值

参　　数	值
Action	android. intent. action. WEB_SEARCH
ExtraKey	query
ExtraValue	greatwall
ActivityPackage	com. google. android. providers. enhancedgooglesearch
ActivityClass	com. google. android. providers. enhancedgooglesearch. Launcher

4. 启动浏览器,并打开指定的网页

启动手机中内置的 Web 浏览器,并打开指定链接地址的网页,需要设置参数 Action 和 DataUri 的值。Action 参数设置为 android. intent. action. VIEW,表示调用手机内部程序浏览指定的内容,但具体会打开哪个程序,还是要根据所指定的“浏览内容”。第二个参数 DataUri 笔者将其设置为 http://android. hrbeu. edu. cn,其实只要出现“http”的字样,Android 系统就会调用系统内部的 Web 浏览器。启动浏览器打开网页的参数值如表 8. 6 所示。

5. 启动谷歌地图,显示指定地点

启动谷歌地图并显示指定地点需要设置的参数依然为 Action 和 DataUri。DataUri 中的参数为“geo: 0,0&q=Potala Palace”,其中“Potala Palace”是要显示的在图上的地点。启动谷歌地图并显示指定地点的参数值设置如表 8. 7 所示。

表 8.6　启动浏览器打开网页的参数值

参　数	值
Action	android. intent. action. VIEW
DataUri	http://android. hrbeu. edu. cn

表 8.7　启动谷歌地图的参数值

参　数	值
Action	android. intent. action. VIEW
DataUri	geo:0,0&q= Potala Palace

下面通过 DreamTour 示例讲解如何在应用程序中启动谷歌地图,搜索并显示指定位置的地图。DreamTour 示例的用户界面如图 8. 24 所示。



图 8. 24　DreamTour 示例的运行界面

界面中“选择景点”下方的 4 个小图片是可以单击的按钮,单击不同的小图片会显示不同的景点名称和图片。4 个小图片分别是复活节岛石像、拉萨布达拉宫、纽约自由女神像和巴黎埃菲尔铁塔。选择不同的景点后,单击“打开地图”按钮可以启动谷歌地图,并自动搜所选择的景点,然后将指定景点的位置信息显示在地图上。

如图 8. 25 所示为 DreamTour 示例界面示意图。

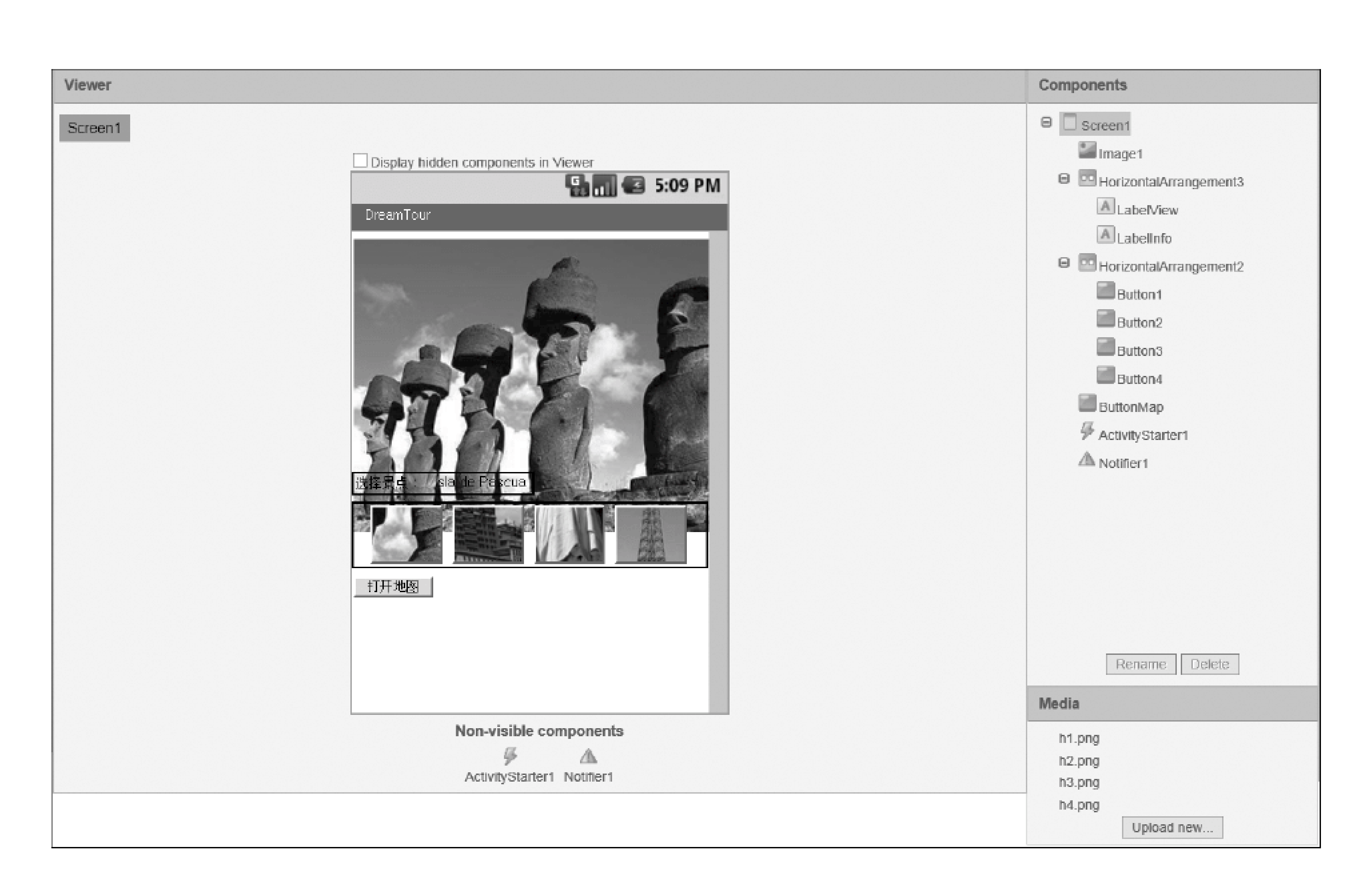


图 8.25 DreamTour 示例界面示意图

需要注意的是，在进行 ActivityStarter 的属性参数设置时，Action 是一定要进行设置的，DataUri 是在模块编辑器中根据用户的选择进行设置的。ActivityClass 和 ActivityPackage 参数可以设置，也可以不设置。如不设置 ActivityClass 和 ActivityPackage 参数，在手机中装有多个地图软件时，会提示用户进行选择，如图 8.26 所示。



图 8.26 完成动作方式选择

如果希望避免从多个地图软件中做选择，可以直接打开谷歌地图，只需要将 ActivityClass 和 ActivityPackage 参数按照表 8.8 进行设置即可。

表 8.8 启动谷歌地图的参数值

参 数	值	类 型
Action	android.intent.action.VIEW	必选
DataUri	geo:0,0&q= Isla de Pascua	必选，模块编辑器中设定
ActivityClass	com.google.android.maps.MapsActivity	可选
ActivityPackage	com.google.android.apps.maps	可选

在模块编辑器中，创建响应 4 个按钮的单击事件，根据用户单击的按钮不同，选择不同的图片在图像 Image1 上显示，并修改标签 LabelInfo 的显示内容，如图 8.27 所示。

在按钮单击事件中，LabelInfo 标签使用的是英文景点名称，景点名称的中英文对照如表 8.9 所示。

在单击 ButtonMap 按钮后，调用 Notifier1 控件的 ShowChooseDialog 方法，打开选择对话框。ButtonMap 单击事件模块如图 8.28 所示。

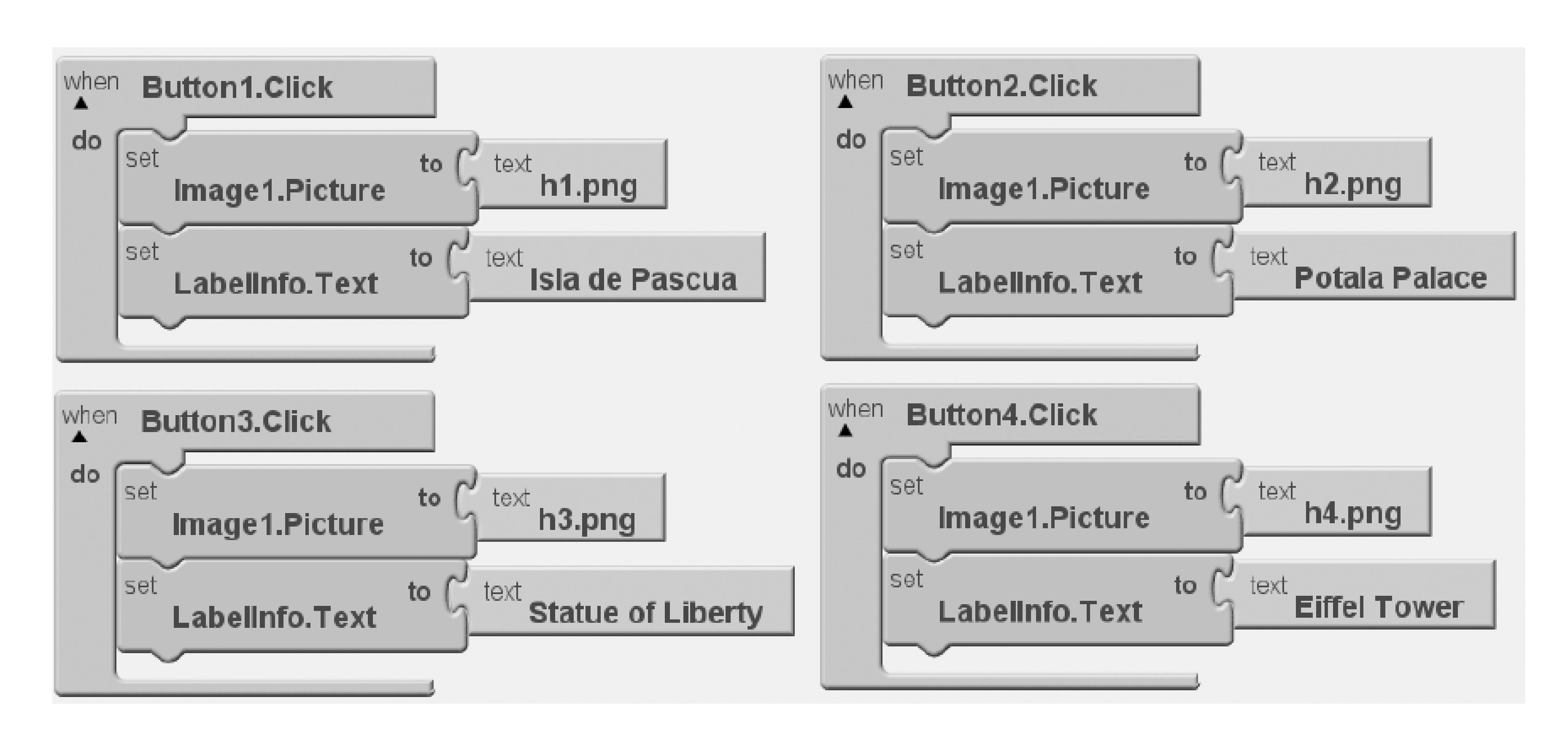


图 8.27 景点按钮单击事件

表 8.9 景点名称的中英文对照

英文名称	中文名称	英文名称	中文名称
Statue of Liberty	纽约自由女神像	Isla de Pascua	复活节岛石像
Potala Palace	拉萨布达拉宫	Eiffel Tower	巴黎埃菲尔铁塔

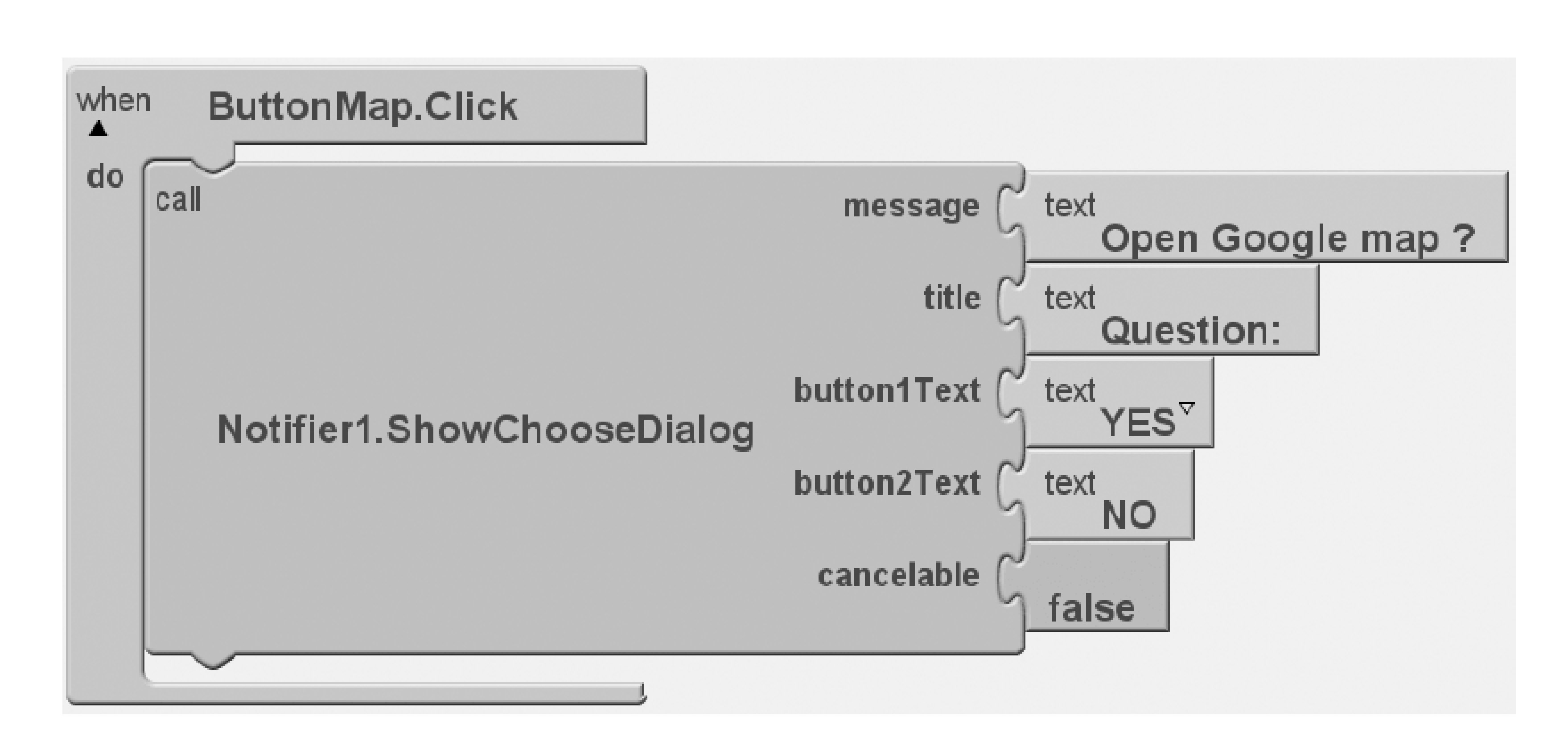


图 8.28 ButtonMap 按钮单击事件

根据 ShowChooseDialog 方法的参数设定，选择对话框的内容如图 8.29 所示。

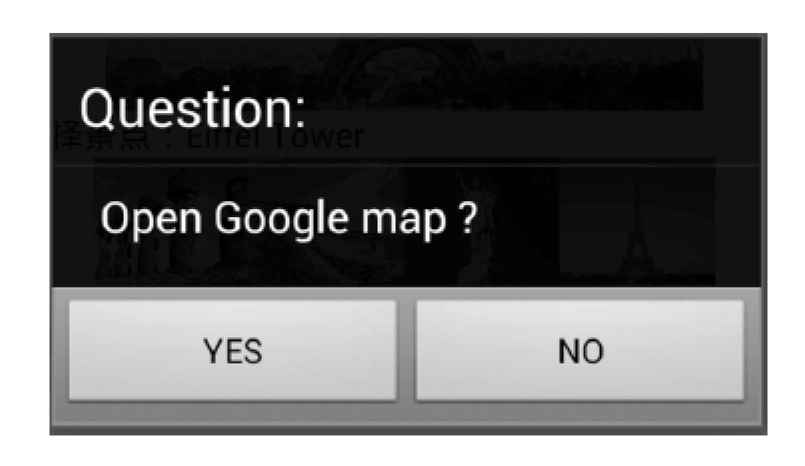


图 8.29 选择对话框

用户在选择对话框中做出抉择后，会触发 Notifier1 控件的 AfterChoosing 事件，根据参数 choice 的值，判断用户的选择。因为用户可以选择的值只有“YES”和“NO”，因此只要判断 choice 的值是否为“YES”。如果是则设置 ActivityStarter1 的 DataUri 属性，然后调用 StartActivity 方法打开谷歌地图，如图 8.30 所示。

DreamTour 示例的全部逻辑模块如图 8.31 所示。

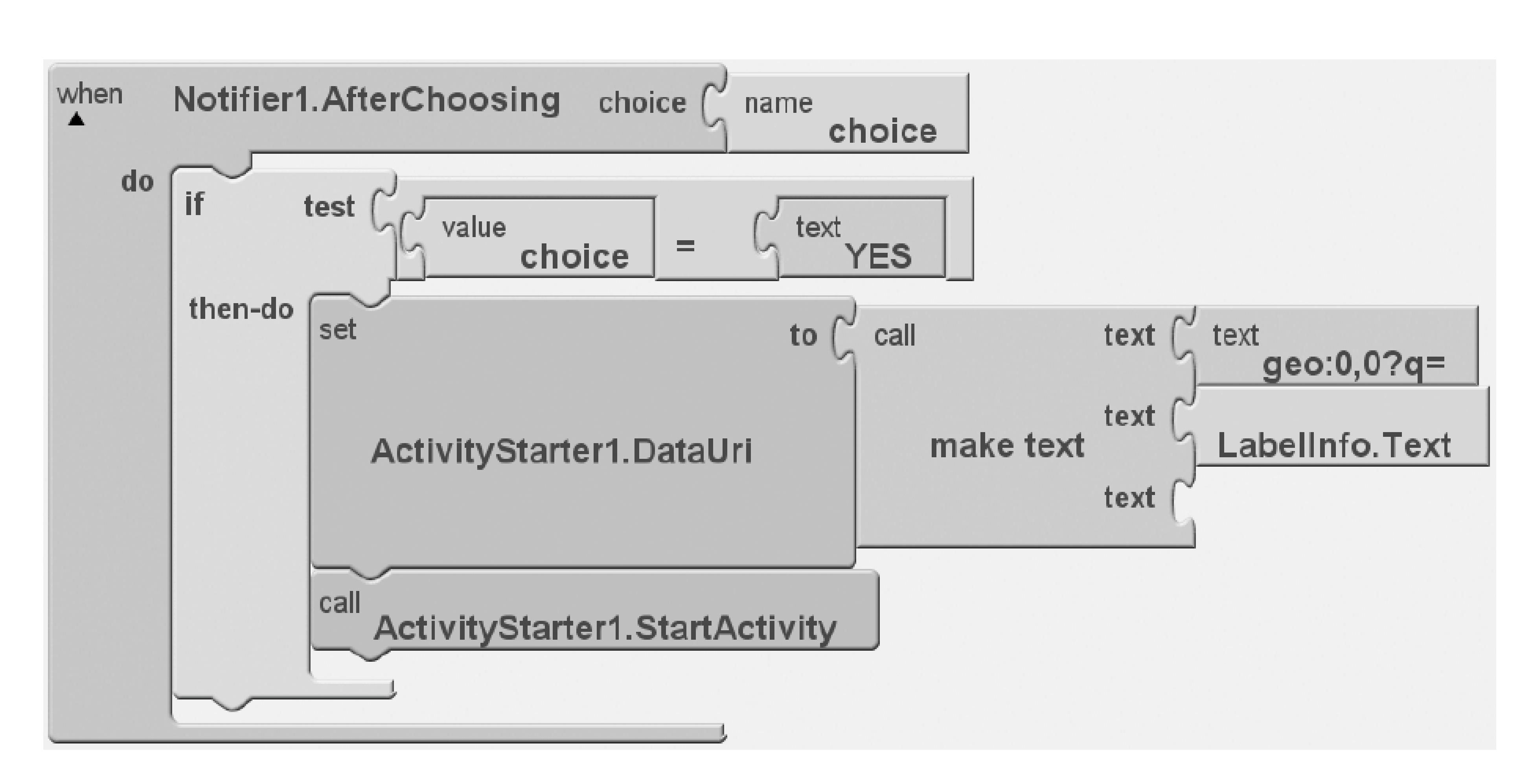


图 8.30　AfterChoosing 事件

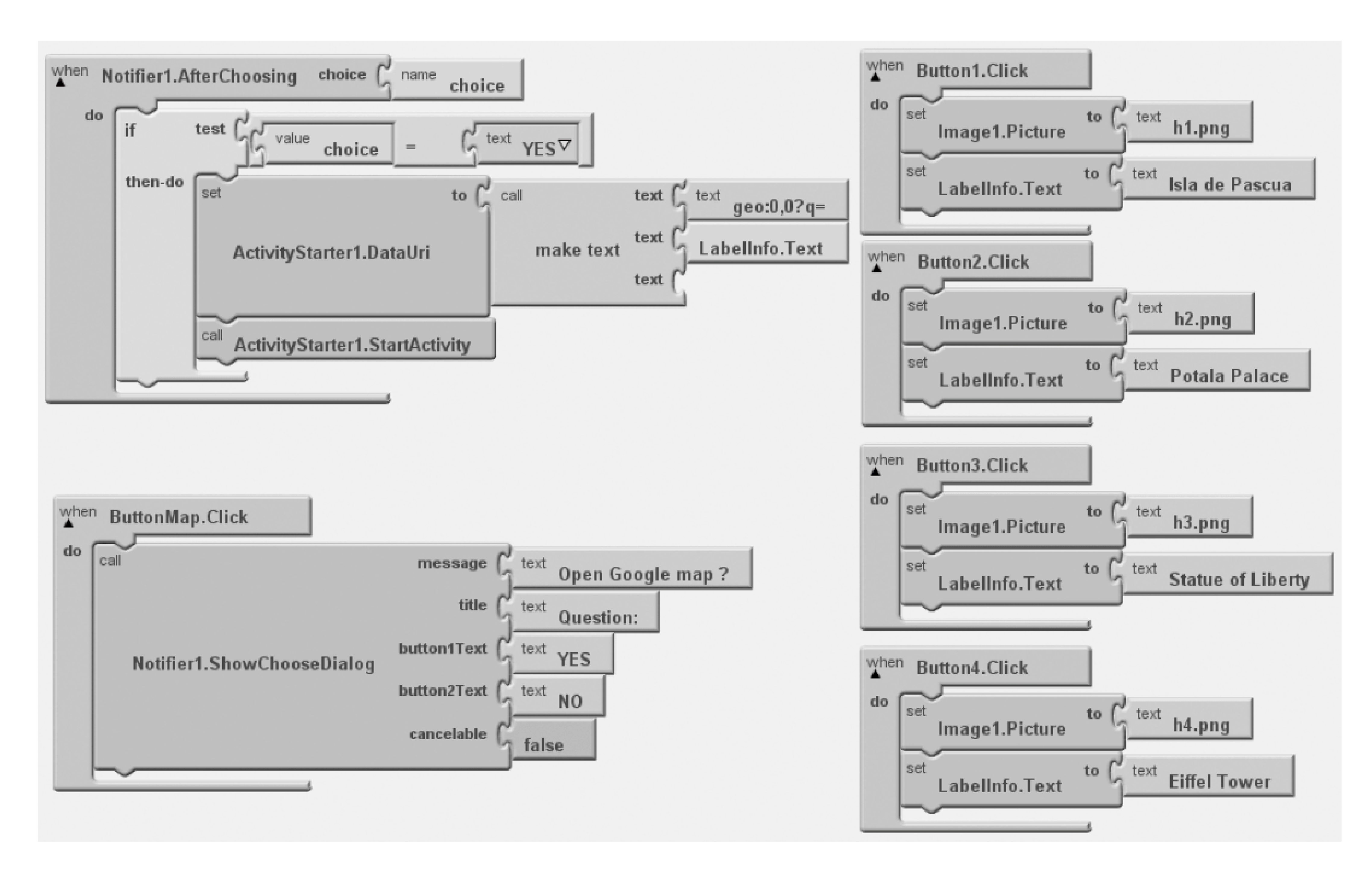


图 8.31　DreamTour 示例的全部逻辑模块

习　　题

1．讨论位置服务和地图应用的发展前景。

2．编程实现轨迹追踪软件。每间隔 60s，同时距离移动大于 1m 的情况下，记录一次位置信息，在数据库中记录 600s 的行动轨迹，并可以尝试在谷歌地图上显示行动轨迹。

附录 A



Build-In(内置)模块

Build-In 中包含 Definition、Text、Lists、Math、Logic、Control 和 Colors 共 7 个模块集，每个模块集的使用说明如附表 A.1～附表 A.7 所示。

A.1 Definition 模块集(定义模块集)

附表 A.1 Definition 模块集使用说明

名　　称	模 块 图 像	说　　明
procedure	to procedure arg do	创建无返回值的函数
procedureWithResult	to procedureWithResult arg do return	创建有返回值的函数
name	name name	自定义的函数参数，与函数的槽 arg 拼接
variable	def variable as	自定义全局变量，可以在程序的任何地方使用
|	|	使用此模块将忽略某些模块的返回值，也可作为连接器

A2 Text 模块集(文本模块集)

附表 A.2 Text 模块集使用说明

名　　称	模块图像	说　　明
text	text text	定义一个字符串型常量
join	join	将指定的两个字符串合成一个新的字符串
make text	call make text text	将所有参数按照从上到下的次序组成一个字符串
length	call length text	获取指定字符串的长度
text<	text< text1 text2	判断字符串 text1 是否小于 text2
text=	text= text1 text2	比较两个字符串是否相等
text>	text> text1 text2	判断字符串 text1 是否大于 text2
trim	trim text	删除指定字符串首尾的空格
upcase	upcase text	将字符串所有字母变为大写字母
downcase	downcase text	将字符串所有字母变为小写字母
starts at	call starts at text piece	获取字符串 piece 在字符串 text 中的位置，若匹配成功则返回 piece 字符的首字母所在位置，若未找到则返回 0
contains	call contains text piece	若字符串 text 中包含字符串 piece，则返回 true，否则返回 false
split at first	call split at first text at	以字符串 at 作为分隔符，将字符串 test 在 at 首次出现的位置分割为两个字符串，并返回分割后的字符串列表。如果 at 不存在，则返回原始字符

续表

名　　称	模块图像	说　　明
split at first of any	call split at first of any text at	at 是分隔符列表,字符串 text 为在首个匹配分隔符列表中任何一个分隔符的位置,将字符串 text 分割为两个字符串,并返回分割后的字符串列表
split	call split text at	利用特定字符将字符串分割为多个字符串,并返回分割后的字符串列表
split at any	call split at any text at	与 split 相同,但可以在 at 处设置多个参数
split at spaces	call split at spaces text	利用空格来分割字符串
segment	call segment text start length	以指定开始点 start 和长度 length 的方式,截取字符串 text 中的一部分字符串
is text empty?	is text empty? text	判断 text 是否为空字符串
replace all	call replace all text segment replacement	将字符串 text 中原字符串(segment)用新字符串(replacement)代替

A3　Lists 模块集(列表模块集)

附表 A.3　Lists 模块集使用说明

模块名称	模　　块	模块说明
make a list	call make a list item	创建一个列表,类表中的元素在槽 item 上添加
select list item	call select list item list index	取出列表 index 索引位置的元素,列表第一个元素的索引号为 1
replace list item	call replace list item list index replacement	用 replacement 元素替代列表 list 中索引位置为 index 的元素

续表

模块名称	模　　块	模块说明
remove list item	call remove list item list index	删除列表中指定索引位置的元素
insert list item	call insert list item list index item	将元素插入到列表中的指定位置
length of list	call length of list list	获取列表中元素的数目，并返回该值
append to list	call append to list list1 list2	将列表 list1 和列表 list2 合成一个新的列表
add items to list	call add items to list list item	在列表最后位置加入指定元素
is in list?	call is in list? thing list	判断元素 thing 是否在列表中，若在则返回 true，不在则返回 false
position in list	call position in list thing list	查找元素(thing)在列表中的索引号，若元素在列表中则返回该索引号，否则返回 0
pick random item	call pick random item list	随机取出列表中的一个元素
is list empty?	call is list empty? list	判断列表是否为空，若为空则返回 true，否则返回 false
copy list	call copy list list	复制列表
is a list?	call is a list? thing	判断指定语句(thing)是否为列表，若是则返回 true，否则返回 false
list to csv row	call list to csv row list	将列表中的元素依次转化为 CSV 表格的一行，并将该行文本内容返回，返回的文本尾部没有换行符

续表

模块名称	模　　块	模块说明
list to csv table	call list to csv table list	将列表中元素以行优先的方式填充到 CSV 表格(有多行),并返回该表格的文本值,表格中每行的元素以逗号隔开,换行时以"\r\n"隔开
list from csv row	call list from csv row text	将 CSV 表格中的一行中每一个值转化为一个列表返回
list from csv table	call list from csv table text	将 CSV 表格中的内容以行优先的次序转化为列表返回,以\n 或 CRFT(\r\n)符号来区分不同的行
lookup in pairs	call lookup in pairs key pairs notFound	返回列表(pairs)中与关键字(key)相关联的值,若未找到返回 notFound 的值

A4 Math 模块集(数学模块集)

附表 A.4 Math 模块集使用说明

模块名称	模　　块	模块说明
number	number 123	定义一个数值型常量
>	>	比较两个数值的大小,若前者大于后者则返回 true,否则返回 false
>=	>=	比较两个数值的大小,若前者大于等于后者返回 true,否则返回 false
<	<	比较两个数值的大小,若前者小于后者则返回 true,否则返回 false
<=	<=	比较两个数值的大小,若前者小于等于后者返回 true,否则返回 false
=	=	判断两数值是否相等,若相等则返回 true,不相等返回 false
not=	not =	判断两数值是否不相等,若不相等则返回 true,否则返回 false
+	+	对两数值进行加法运算,并返回结果

续表

模块名称	模　　块	模块说明
—	-	对两数值进行减法运算，并返回结果
×	×	对两数值进行乘法运算，并返回结果
/	/	对两数值进行除法运算，并返回结果
sqrt	call sqrt	对指定数值进行平方根运算，并返回结果
random fraction	call random fraction	返回一个在 0 和 1 之间的随机小数
random integer	call random integer from to	返回一个指定范围内的随机整数，通过 from 和 to 设置随机整数的取值范围
random set seed	call random set seed seed	指定一个数字为随机数生成器的种子
negate	call negate	返回给定数字的相反数
min	call min	返回给定多个数字中的最小值
max	call max	返回给定多个数字中的最大值
quotient	call quotient	返回第一个数字除以第二个数字的商的整数部分
remainder	call remainder	返回第一个数字除以第二个数字的余数，余数与第一个数字同号，例如 remainder(13，−4)＝1
modulo	call modulo	返回第一个数字除以第二个数字的余数，但是要求该余数与第二个数字同号，例如 module(13，−4)＝−3
abs	call abs	返回给定数字的绝对值
round	call round	对给定数字进行四舍五入运算，返回值为整数
floor	call floor	对给定数字进行舍小数部分运算，返回值为整数

续表

模块名称	模块	模块说明
ceiling	call ceiling	对给定数字进行无条件进位运算,返回值为整数
expt exponent	call base expt exponent	指数运算,base 为指数的基数,exponent 为指数的幂
exp	call exp	计算以 e(2.71828…)为基数的指数
log	call log	计算给定数字的自然对数
sin	call sin degrees	计算正弦值
cos	call cos degrees	计算余弦值
tan	call tan degrees	计算正切值
asin	call asin	计算反正弦值
acos	call acos	计算反余弦值
atan	call atan	计算反正切值
atan2	call y atan2 x	计算 y/x 值的反正切值
convert radians to degrees	call convert radians to degrees radians	将弧度转化为角度
convert degrees to radians	call convert degrees to radians degrees	将角度转化为弧度
format as decimal	call number format as decimal places	将数字(number)转化为有指定小数位数(places)的数值,若小数位过多则进行四舍五入,不够则补 0
is a number?	call is a number? thing	判断参数 thing 是否为数值,若是则返回 true,否则返回 false

A5 Logic模块集(逻辑运算模块集)

附表 A.5 Logic 模块集使用说明

模块名称	模 块	模块说明
true	true	布尔值为真
false	false	布尔值为假
not	not	逻辑非运算
=	=	判断两参数是否相等,若两参数为数字,判断其数值是否相等;若是字符,则必须完全匹配才相等(区分大小写)
and	and test	逻辑与运算,若测试条件中有一个为假,则运算结果为假(返回 false),否则为真(返回 true)
or	or test	逻辑或运算,若测试条件中有一个为真,则运算结果为真(返回 true),否则为假(返回 false)

A6 Control指令集(控制指令集)

附表 A.6 Control 指令集使用说明

模块名称	模 块	模块说明
if	if test then-do	判断语句,若指定条件 test 的值为 true,则执行 then-do 区域中的语句,否则跳过此段
ifelse	ifelse test then-do else-do	判断语句,若指定条件 test 的值为 true,则执行 then-do 区域中的语句,否则执行 else-do 区域中的语句

续表

模块名称	模　　块	模块说明
choose	choose test then-do then-return else-do else-return	判断语句，若指定条件 test 的值为 true，则执行 then-do 区域中的语句，并返回 then-return 的值；否则，执行 else-do 区域中的语句，并返回 else-return 的值
foreach	foreach variable do in list	循环语句，循环次数为列表中元素的个数，变量 variable 次序取列表中元素的值，执行 do 区域中的语句
for range	for range variable start end step do	循环语句，取介于 start 和 end 之间且间隔为 step 的值，将这些值依次传递给 variable 执行 do 区域中的语句
while	while test do	循环语句，当指定条件 test 的值为 true，则执行 do 中的语句，直到 test 值为 false，跳出循环
open another screen	call open another screen screenName	打开另一个屏幕页，屏幕页的名称由 screenName 参数指定
open another screen with start value	call open another screen with start value screenName startValue	打开另一个屏幕页，屏幕页的名称由 screenName 参数指定，并向屏幕页中传递一个初始化参数
get start value	call get start value	获取屏幕页的一个初始化参数，若无则返回一个为空的字符
close screen	call close screen	关闭当前的屏幕页
close screen with value	call close screen with value result	关闭当前屏幕页，并向父界面返回一个参数

续表

模块名称	模块	模块说明
get plain start text	call get plain start text	当一个应用的某一屏幕页被其他应用打开时，向该窗口传递一个初始化该屏幕页的字符串
close screen with plain text	call close screen with plain text text	关闭当前屏幕页，并返回一个字符串给父屏幕页
close application	call close application	关闭当前应用程序

A7 Colors 模块集(色彩模块集)

在该模块集中，系统提供了 14 种预定义的颜色，用户也可以自定义颜色模块。App Inventor 中的颜色是以编码形式存在于系统中，每种颜色由 4 个参数唯一确定，每个参数的取值范围都是 0～255。前三个参数代表红(R)、绿(G)、蓝(B)三原色的强度，第四个参数为不透明度(Opacity)。

附表 A.7　Colors 模块集使用说明

模块名称	模块	模块说明
make color	call make color components	创建一个内建颜色库中没有的颜色，components 为创建该颜色的参数列表，列表中应有 4 个元素
split color	call split color color	将该颜色(color)分解为创建该颜色的 4 个参数
None	color None	无色
Black	color Black	黑色
Blue	color Blue	蓝色
Cyan	color Cyan	青色
Dark Gray	color Dark Gray	深灰色

续表

模块名称	模　　块	模块说明
Gray	color Gray	灰色
Green	color Green	绿色
Light Gray	color Light Gray	浅灰色
Magenta	color Magenta	洋红色
Orange	color Orange	橙色
Pink	color Pink	粉红色
Red	color Red	红色
White	color White	白色
Yellow	color Yellow	黄色

附录 B



控 件 简 介

B.1　Basic(基本)控件

基本控件共 10 个，包括 Button、Canvas、CheckBox、Clock、Image、Label、ListPicker、PasswordTextBox、TextBox 和 TinyDB，各个控件的事件、属性和方法如附表 B. 1～附表 B. 10 所示。

1. Button(按钮)

Button 是界面上最基本的控件，主要提供单击式的触发操作，实现最基本的人机交互功能。

附表 B. 1　Button 控件说明

模 块 名 称	模块及模块说明	
Click(事件)	when Button1.Click do	按钮单击事件
LongClick(事件)	when Button1.LongClick do	按钮长时间单击事件
GotFocus(事件)	when Button1.GotFocus do	获取焦点事件
LostFocus(事件)	when Button1.LostFocus do	失去焦点事件

续表

模块名称	模块及模块说明
BackgroundColor(属性)	set Button1.BackgroundColor to Button1.BackgroundColor 按钮背景色
Enabled(属性)	set Button1.Enabled to Button1.Enabled 按钮可用性
Image(属性)	set Button1.Image to Button1.Image 按钮的背景图片
Text(属性)	set Button1.Text to Button1.Text 按钮显示的文本
TextColor(属性)	set Button1.TextColor to Button1.TextColor 按钮的文本颜色
Visible(属性)	set Button1.Visible to Button1.Visible 按钮的可见性
Width(属性)	set Button1.Width to Button1.Width 按钮的宽度
Height(属性)	set Button1.Height to Button1.Height 按钮的高度
Button1(实例)	component Button1 按钮的实例

2. Canvas(画布)

Canvas是一种可在其上绘制图像的控件,除了作为绘制图形的承载体以外,还经常作为游戏的背景画面。

附表 B.2 Canvas 控件说明

模块名称	模块及模块说明	
Dragged(事件)	when Canvas1.Dragged startX startY prevX prevY currentX currentY draggedSprite do	拖曳事件， startX、startY：表示拖曳起始位置的坐标值； prevX、prevY：表示上一Dragged事件的坐标值； currentX、currentY：表示当前拖曳点的坐标值； draggedSprite：判断精灵是否被拖曳
Flung(事件)	when Canvas1.Flung x y speed heading xvel yvel flungSprite do	快速滑动事件， x、y：滑动事件初始坐标； speed：滑动事件的速度(单位为像素/毫秒)； heading：滑动事件的角度(0～360)； xvel、yvel：速度在 X 轴和 Y 轴的分量； flungSprite：判断滑动起始点附近的精灵
Touched(事件)	when Canvas1.Touched x y touchedSprite do	触碰事件， x、y：触控点的 X、Y 轴坐标值； touchedSprite：判断触控事件是否被触发
TouchDown(事件)	when Canvas1.TouchDown x y do	按下事件， x、y：手指按下时触控点的 X、Y 轴坐标值
TouchUp(事件)	when Canvas1.TouchUp x y do	抬起事件， x、y：手指抬起时触碰点的 X、Y 轴坐标值
Clear(方法)	call Canvas1.Clear	清空画布元素，如果画布上已设置图片，此图片会被清除

续表

模块名称	模块及模块说明
DrawCircle(方法)	call Canvas1.DrawCircle x y r 在画布上绘制圆形图案,其中,x 和 y 为圆心坐标,r 为半径
DrawLine(方法)	call Canvas1.DrawLine x1 y1 x2 y2 在画布上从(x1,y1)点到(x2,y2)点绘制直线
DrawPoint(方法)	call Canvas1.DrawPoint x y 在画布上绘制圆点图案,位置坐标为(x,y)
DrawText(方法)	call Canvas1.DrawText text x y 在画布上坐标为(x,y)的位置显示文本 text 的内容
DrawTextAtAngle(方法)	call Canvas1.DrawTextAtAngle text x y angle 在画布上坐标为(x,y)的位置以 angle 角度显示文本 text 的内容
GetBackgroundPixelColor(方法)	call Canvas1.GetBackgroundPixelColor x y 获取画布上坐标为(x,y)点的背景颜色
GetPixelColor(方法)	call Canvas1.GetPixelColor x y 获取画布上坐标为(x,y)点的颜色
SaveAs(方法)	call Canvas1.SaveAs fileName 将画布当前状态截图存储在 SD 卡上,文件名为 fileName(只能是 JPEG、JPG、PNG 文件),并返回该文件的完整存储路径
Save(方法)	call Canvas1.Save 将画布当前状态的截图存储在 SD 卡上,并返回该文件完整的存储路径

续表

模块名称	模块及模块说明
BackgroundColor(属性)	set Canvas1.BackgroundColor to Canvas1.BackgroundColor 画布背景色
BackgroundImage(属性)	set Canvas1.BackgroundImage to Canvas1.BackgroundImage 画布背景图片
FontSize(属性)	set Canvas1.FontSize to Canvas1.FontSize 画布字体大小
LineWidth(属性)	set Canvas1.LineWidth to Canvas1.LineWidth 画笔宽度(绘制直线时)
PaintColor(属性)	set Canvas1.PaintColor to Canvas1.PaintColor 画笔颜色
Visible(属性)	set Canvas1.Visible to Canvas1.Visible 画布可见性
Width(属性)	set Canvas1.Width to Canvas1.Width 画布的宽度
Height(属性)	set Canvas1.Height to Canvas1.Height 画布的高度
Canvas1(实例)	component Canvas1 画布的一个实例

3. CheckBox(复选框)

CheckBox是可以同时选中多项的选项框,供用户在不同选项间进行多项选择时使用,在程序当中起到条件识别的作用。

附表 B.3 CheckBox 控件说明

模块名称	模块及模块说明
Changed(事件)	when CheckBox1.Changed do 复选框状态改变事件(选中或者取消选中)
GotFocus(事件)	when CheckBox1.GotFocus do 获取焦点事件
LostFocus(事件)	when CheckBox1.LostFocus do 失去焦点事件
BackgroundColor(属性)	set CheckBox1.BackgroundColor to CheckBox1.BackgroundColor 复选框的背景颜色
Checked(属性)	set CheckBox1.Checked to CheckBox1.Checked 复选框是否被选中
Enabled(属性)	set CheckBox1.Enabled to CheckBox1.Enabled 复选框是否可用
FontSize(属性)	set CheckBox1.FontSize to CheckBox1.FontSize 复选框字体大小
Text(属性)	set CheckBox1.Text to CheckBox1.Text 复选框的文字
TextColor(属性)	set CheckBox1.TextColor to CheckBox1.TextColor 复选框的文本颜色
Visible(属性)	set CheckBox1.Visible to CheckBox1.Visible 复选框是否可见



续表

模块名称	模块及模块说明
Width(属性)	set CheckBox1.Width to CheckBox1.Width 复选框的宽度
Height(属性)	set CheckBox1.Height to CheckBox1.Height 复选框的高度
CheckBox1(实例)	component CheckBox1 复选框的一个实例

4. Clock(时钟)

Clock 是非可视化组件,可以获取当前时间、格式化输出时间、对时间进行运算,还可以在固定的时间间隔触发事件。

附表 B.4　Clock 控件说明

模块名称	模块及模块说明	
Timer(事件)	when Clock1.Timer do	时钟触发事件
SystemTime(方法)	call Clock1.SystemTime	手机的内部系统时间,单位为微秒
Now(方法)	call Clock1.Now	从手机时钟读取的当前时间
MakeInstant(方法)	call Clock1.MakeInstant from	通过"月/日/年 时:分:秒"、"月/日/年"、"时:分"的格式定义时间点
MakeInstantFromMillis(方法)	call Clock1.MakeInstantFromMillis millis	通过毫秒数定义时间点
GetMillis(方法)	call Clock1.GetMillis instant	从 1970 年 1 月 1 日开始累计到现在的时间,单位为毫秒
AddSeconds(方法)	call Clock1.AddSeconds instant seconds	计算 instant 上增加若干秒以后的时间点

续表

模块名称	模块及模块说明	
AddMinutes(方法)	call Clock1.AddMinutes instant minutes	计算 instant 上增加若干分钟以后的时间点
AddHours(方法)	call Clock1.AddHours instant hours	计算 instant 增加若干小时以后的时间点
AddDays(方法)	call Clock1.AddDays instant days	计算 instant 增加若干天以后的时间点
AddWeeks(方法)	call Clock1.AddWeeks instant weeks	计算 instant 增加若干周以后的时间点
AddMonths(方法)	call Clock1.AddMonths instant months	计算 instant 增加若干月以后的时间点
AddYears(方法)	call Clock1.AddYears instant years	计算 instant 增加若干年以后的时间点
Duration(方法)	call Clock1.Duration start end	计算两个时间点 start 和 end 的时间间隔,单位为毫秒
Second(方法)	call Clock1.Second instant	获取 instant 时间点的秒数
Minute(方法)	call Clock1.Minute instant	获取 instant 时间点的分钟数
Hour(方法)	call Clock1.Hour instant	获取 instant 时间点的小时数
DayOfMonth(方法)	call Clock1.DayOfMonth instant	获取 instant 时间点的日期,范围为 1～31 的数字
Weekday(方法)	call Clock1.Weekday instant	获取 instant 时间点是周几,范围为 1(周日)～7(周六)的数字
WeekdayName(方法)	call Clock1.WeekdayName instant	获取 instant 时间点是周几的名称
Month(方法)	call Clock1.Month instant	获取 instant 时间点的月份数,范围为 1～12 的数字

续表

模块名称	模块及模块说明
MonthName(方法)	call Clock1.MonthName instant 获取instant时间点的月份,用名称表述
Year(方法)	call Clock1.Year instant 获取instant时间点的年份
FormatDateTime(方法)	call Clock1.FormatDateTime instant 格式化输出instant时间点的日期和时间
FormatDate(方法)	call Clock1.FormatDate instant 格式化输出instant时间点的日期
FormatTime(方法)	call Clock1.FormatTime instant 格式化输出instant时间点的时间
TimerAlwaysFires(属性)	set Clock1.TimerAlwaysFires to Clock1.TimerAlwaysFires 设置时钟是否多次触发
TimerInterval(属性)	set Clock1.TimerInterval to Clock1.TimerInterval 设置时间定时器的时间间隔
TimerEnabled(属性)	set Clock1.TimerEnabled to Clock1.TimerEnabled 时钟是否可用
Clock1(实例)	component Clock1 时钟的一个实例

5. Image(图像)

Image控件用于在界面上显示各种图像文件,不支持事件。

附表B.5　Image控件说明

模块名称	模块及模块说明
Animation(属性)	set Image1.Animation to 设置动态图片
Picture(属性)	set Image1.Picture to Image1.Picture 图片背景

续表

模块名称	模块及模块说明
Visible(属性)	set Image1.Visible to Image1.Visible 图片可见性
Width(属性)	set Image1.Width to Image1.Width 图片的宽度
Height(属性)	set Image1.Height to Image1.Height 图片的高度
Image1(实例)	component Image1 图片的一个实例

6. Label(标签)

标签主要起到文字显示的作用,但标签不允许用户进行输入操作,只能够显示文字信息。

附表 B.6 Label 控件说明

模块名称	模块及模块说明
BackgroundColor(属性)	set Label1.BackgroundColor to Label1.BackgroundColor 标签背景颜色
FontSize(属性)	set Label1.FontSize to Label1.FontSize 标签字体大小
Text(属性)	set Label1.Text to Label1.Text 标签栏里显示的文字
TextColor(属性)	set Label1.TextColor to Label1.TextColor 标签的文字颜色

续表

模 块 名 称	模块及模块说明
Visible(属性)	set Label1.Visible to Label1.Visible 标签可见性
Width(属性)	set Label1.Width to Label1.Width 标签的宽度
Height(属性)	set Label1.Height to Label1.Height 标签的高度
Label1(实例)	component Label1 标签的一个实例

7. ListPicker(选项列表)

ListPicker 是从多个选项中选取某一个选项的控件,适合多选一的情况。

附表 B.7 ListPicker 控件说明

模 块 名 称	模块及模块说明	
BeforePicking(事件)	when ListPicker1.BeforePicking do	选前事件,当点开选项列表,但没有选择其中的某项时产生
AfterPicking(事件)	when ListPicker1.AfterPicking do	选后事件,当点开选项列表并选中其中的某项时产生
GotFocus(事件)	when ListPicker1.GotFocus do	获取焦点事件
LostFocus(事件)	when ListPicker1.LostFocus do	失去焦点事件
Open(方法)	call ListPicker1.Open	打开选项列表供选择

续表

模块名称	模块及模块说明
BackgroundColor(属性)	set ListPicker1.BackgroundColor to ImagePicker1.BackgroundColor 选项列表的背景颜色
Elements(属性)	set ListPicker1.Elements to ListPicker1.Elements 选项列表的元素内容
ElementsFromString(属性)	set ListPicker1.ElementsFromString to 从 to 中导入选项列表的元素内容
Enabled(属性)	set ListPicker1.Enabled to ListPicker1.Enabled 选项列表可用性
Image(属性)	set ListPicker1.Image to ListPicker1.Image 选项列表的图片
Selection(属性)	set ListPicker1.Selection to ListPicker1.Selection 用户选中的列表项属性
SelectionIndex(属性)	set ListPicker1.SelectionIndex to ListPicker1.SelectionIndex 用户选中的列表项序号
Text(属性)	set ListPicker1.Text to ListPicker1.Text 选项列表显示的文本内容
TextColor(属性)	set ListPicker1.TextColor to ListPicker1.TextColor 选项列表的文本颜色
Visible(属性)	set ListPicker1.Visible to ListPicker1.Visible 选项列表的可见性

续表

模块名称	模块及模块说明
Width(属性)	set ListPicker1.Width to / ListPicker1.Width 选项列表的宽度
Height(属性)	set ListPicker1.Height to / ListPicker1.Height 选项列表的高度
ListPicker1(实例)	component ListPicker1 选项列表的一个实例

8. PasswordTextBox(密码框)

PasswordTextBox是一种特殊的文本框,一般用于接受用户输入密码,用户输入密码时,密码框会对输入的内容进行屏蔽处理。

附表 B.8　PasswordTextBox 控件说明

模块名称	模块及模块说明	
GotFocus(事件)	when PasswordTextBox1.GotFocus do	获取焦点事件
LostFocus(事件)	when PasswordTextBox1.LostFocus do	失去焦点事件
Background Color(属性)	set PasswordTextBox1.BackgroundColor to / PasswordTextBox1.BackgroundColor 密码框的背景色	
Enabled(属性)	set PasswordTextBox1.Enabled to / PasswordTextBox1.Enabled 密码框是否可用	
FontSize(属性)	set PasswordTextBox1.FontSize to / PasswordTextBox1.FontSize 密码框字体大小	

续表

模块名称	模块及模块说明
Hint(属性)	set PasswordTextBox1.Hint to PasswordTextBox1.Hint 密码框的提示信息
Text(属性)	set PasswordTextBox1.Text to PasswordTextBox1.Text 密码框里显示的文字
TextColor(属性)	set PasswordTextBox1.TextColor to PasswordTextBox1.TextColor 文本的颜色
Visible(属性)	set PasswordTextBox1.Visible to PasswordTextBox1.Visible 密码框是否可见
Width(属性)	set PasswordTextBox1.Width to PasswordTextBox1.Width 密码框的宽度
Height(属性)	set PasswordTextBox1.Height to PasswordTextBox1.Height 密码框的高度
PasswordTextBox1(实例)	component PasswordTextBox1 密码框的一个实例

9. TextBox(文本框)

TextBox 是一种供用户进行输入文字的容器。虽然文本框可以显示文字信息,但其主要的功能还是为用户提供输入信息的区域,比如登录框、搜索栏或是编辑文字的写字板等。

附表 B.9 TextBox 控件说明

模块名称	模块及模块说明	
GotFocus(事件)	when TextBox1.GotFocus do	获取焦点事件

续表

模块名称	模块及模块说明
LostFocus(事件)	when TextBox1.LostFocus do　　失去焦点事件
HideKeyboard(方法)	call TextBox1.HideKeyboard　　隐藏软键盘
BackgroundColor(属性)	set TextBox1.BackgroundColor to TextBox1.BackgroundColor 文本框的背景颜色
Enabled(属性)	set TextBox1.Enabled to TextBox1.Enabled 文本框的可用性
FontSize(属性)	set TextBox1.FontSize to TextBox1.FontSize 文本框的字体大小
MultiLine(属性)	set TextBox1.MultiLine to TextBox1.MultiLine 文本框是否支持多行输入
NumbersOnly(属性)	set TextBox1.NumbersOnly to TextBox1.NumbersOnly 文本框是否只允许输入数字
Hint(属性)	set TextBox1.Hint to TextBox1.Hint 文本框的提示信息
Text(属性)	set TextBox1.Text to TextBox1.Text 文本框的显示文字
TextColor(属性)	set TextBox1.TextColor to TextBox1.TextColor 文本框内文字的颜色

续表

模块名称	模块及模块说明
Visible(属性)	set TextBox1.Visible to / TextBox1.Visible 文本框的可见性
Height(属性)	set TextBox1.Height to / TextBox1.Height 文本框的高度
Width(属性)	set TextBox1.Width to / TextBox1.Width 文本框的宽度
TextBox1(实例)	component TextBox1 文本框的一个实例

10. TinyDB(微型数据库)

TinyDB是一个微型数据库,提供基于标签(关键字)的数据的存储和读取功能。

附表 B.10 TinyDB控件说明

模块名称	模块及模块说明	
StoreValue(方法)	call TinyDB1.StoreValue tag valueToStore	数据存储,tag是标签,valueToStore是要存储的数据,可以是字符串或列表
GetValue(方法)	call TinyDB1.GetValue tag	数据读取,根据标签tag获取数据
TinyDB1(实例)	component TinyDB1	微型数据库的一个实例

B.2 Media(媒体)控件

媒体控件共6个,包括Camcorder、Camera、ImagePicker、Player、Sound和VideoPlayer,各个控件的事件、属性和方法如附表B.11～附表B.16所示。

1. Camcorder(录像机)

Camcorder主要功能是利用手机的摄像头实现视频的录制。

附表 B.11　Camcorder 控件说明

模块名称	模块及模块说明	
AfterRecording(事件)	when Camcorder1.AfterRecording clip name clip do	录像结束后触发的事件，clip 代表视频的存储路径
RecordVideo(方法)	call Camcorder1.RecordVideo	启动手机摄像头的录制功能
Camcorder 1(实例)	component Camcorder1	录像控件的一个实例

2. Camera(相机)

Camera 主要是实现手机的拍照功能。

附表 B.12　Camera 控件说明

模块名称	模块及模块说明	
AfterPicture(事件)	when Camera1.AfterPicture image name image do	拍照后事件，image 是相片的存储路径
TakePicture(方法)	call Camera1.TakePicture	相机拍照
Camera1(实例)	component Camera1	相机的一个实例

3. ImagePicker(选图工具)

ImagePicker 的主要功能是从手机相册的图片库中选取图片。

附表 B.13　ImagePicker 控件说明

模块名称	模块及模块说明	
AfterPicking(事件)	when ImagePicker1.AfterPicking do	选图后事件
BeforePicking(事件)	when ImagePicker1.BeforePicking do	选图前事件

续表

模块名称	模块及模块说明
GotFocus(事件)	when ImagePicker1.GotFocus do　获取焦点事件
LostFocus(事件)	when ImagePicker1.LostFocus do　失去焦点事件
Open(方法)	call ImagePicker1.Open　打开选图工具浏览图片
BackgroundColor(属性)	set ImagePicker1.BackgroundColor to　ImagePicker1.BackgroundColor 选图工具的背景颜色
Enabled(属性)	set ImagePicker1.Enabled to　ImagePicker1.Enabled 选图工具是否可用
Height(属性)	set ImagePicker1.Height to　ImagePicker1.Height 选图工具的高度
Image(属性)	set ImagePicker1.Image to　ImagePicker1.Image 选图工具的背景图片
Text(属性)	set ImagePicker1.Text to　ImagePicker1.Text 选图工具的显示文字
TextColor(属性)	set ImagePicker1.TextColor to　ImagePicker1.TextColor 选图工具的文字颜色
Visible(属性)	set ImagePicker1.Visible to　ImagePicker1.Visible 选图工具的可见性
Width(属性)	set ImagePicker1.Width to　ImagePicker1.Width 选图工具的宽度

续表

模块名称	模块及模块说明
Selection(属性)	ImagePicker1.Selection 所选取图片的路径
ImagePicker1(实例)	component ImagePicker1 选图工具的一个实例

4. Player(音频播放器)

Player 用来播放音频文件,一般用于播放时间较长的音频文件,如音乐、录音等。

附表 B.14 Player 控件说明

模块名称	模块及模块说明	
Completed(事件)	when Player1.Completed do	播放完毕事件
Pause(方法)	call Player1.Pause	暂停播放
Start(方法)	call Player1.Start	开始播放
Stop(方法)	call Player1.Stop	停止播放
Vibrate(方法)	call Player1.Vibrate milliseconds	使手机振动,milliseconds 是振动时间,单位为毫秒
Volume(属性)	set Player1.Volume to	设置播放的音量
IsPlaying(属性)	Player1.IsPlaying	是否正在播放
Source(属性)	set Player1.Source to Player1.Source 播放的文件	

续表

模块名称	模块及模块说明
IsLooping(属性)	set Player1.IsLooping to / Player1.IsLooping 是否循环播放
Player1(实例)	component Player1 音频播放器的一个实例

5. Sound(音效播放器)

Sound 一般可用来播放较短的音频文件或者控制手机的振动。

附表 B.15 Sound 控件说明

模块名称	模块及模块说明
Pause(方法)	call Sound1.Pause　暂停播放
Play(方法)	call Sound1.Play　开始播放
Resume(方法)	call Sound1.Resume　暂停播放后继续播放
Stop(方法)	call Sound1.Stop　停止播放
Vibrate(方法)	call Sound1.Vibrate millisecs　使手机振动,millisecs 是振动时间,单位为毫秒
MinimumInterval(属性)	set Sound1.MinimumInterval to / Sound1.MinimumInterval 播放最短时间间隔
Source(属性)	set Sound1.Source to / Sound1.Source 播放的音效源
Sound1(实例)	component Sound1 音效播放器的一个实例

6. VideoPlayer(视频播放器)

VideoPlayer 主要用于播放视频文件,提供基础的播放控制功能,包括播放、暂停、调整播放位置等。

附表 B. 16 VideoPlayer 控件说明

模块名称	模块及模块说明
Completed(事件)	when VideoPlayer1.Completed do　播放完毕事件
GetDuration(方法)	call VideoPlayer1.GetDuration　视频长度
Pause(方法)	call VideoPlayer1.Pause　暂停播放
SeekTo(方法)	call VideoPlayer1.SeekTo ms　调整播放位置到所指定时间处
Start(方法)	call VideoPlayer1.Start　开始播放
Source(属性)	set VideoPlayer1.Source to　播放文件的路径
FullScreen(属性)	set VideoPlayer1.FullScreen to　VideoPlayer1.FullScreen 视频播放器是否全屏
Height(属性)	set VideoPlayer1.Height to　VideoPlayer1.Height 视频播放器的高度
Visible(属性)	set VideoPlayer1.Visible to　VideoPlayer1.Visible 视频播放器是否可见
Width(属性)	set VideoPlayer1.Width to　VideoPlayer1.Width 视频播放器的宽度
VideoPlayer1(实例)	component VideoPlayer1 视频播放器的一个实例

B.3 Animation(动画)控件

动画控件共两个，包括 Ball 和 ImageSprite，各个控件的事件、属性和方法如附表 B.17 和附表 B.18 所示。

1. Ball(球体)

Ball 是球形状的精灵，可根据属性进行移动，可对其进行触摸和拖曳操作，也可以与其他的精灵(图像精灵或其他球)及画布边缘进行交互。

附表 B.17 Ball 控件说明

模块名称	模块及模块说明	
CollidedWith(事件)	when Ball1.CollidedWith other do	碰撞事件，球体与其他精灵发生碰撞的事件
Dragged(事件)	when Ball1.Dragged startX startY prevX prevY currentX currentY do	拖曳事件，当手指在画布上拖曳球体时触发本事件。 startX、startY：表示拖曳事件开始时 X、Y 坐标值； prevX、prevY：表示上一个拖曳事件产生时的 X、Y 坐标值； currentX、currentY：表示产生该事件时的 X、Y 坐标轴数值
Flung(事件)	when Ball1.Flung x y speed heading xvel yvel do	滑动事件，只有手指在画布上快速滑动时触发本事件。 x、y：滑动事件初始坐标； speed：滑动事件的速度(单位为像素/毫秒)； heading：滑动事件的角度(0～360)； xvel、yvel：速度在 X 轴和 Y 轴的分量
EdgeReached(事件)	when Ball1.EdgeReached edge do	触壁事件，当球触碰画布边缘时触发本事件，并返回所到达边缘的位置信息，其中 edge 值是返回的边缘位置信息

续表

模块名称	模块及模块说明	
NoLonger-CollidingWith(事件)	when Ball1.NoLongerCollidingWith other do	不再碰撞事件,当球体与其他精灵从碰撞状态分离时触发本事件
Touched(事件)	when Ball1.Touched x y do	触摸事件,当用户对球进行触摸时触发本事件。 x、y:触控点的 X、Y 轴坐标值
TouchDown(事件)	when Ball1.TouchDown x y do	按下事件,当用户对球进行按下操作时触发本事件。 x、y:手指按下时触摸点的 X、Y 轴坐标值
TouchUp(事件)	when Ball1.TouchUp x y do	抬起事件,当用户对球进行按下操作后抬起时触发本事件。 x、y:手指抬起时触摸点的 X、Y 轴坐标值
Bounce(方法)	call Ball1.Bounce edge	球体反弹,一般在球体与精灵碰撞后使用
CollidingWith(方法)	call Ball1.CollidingWith other	返回球体是否与指定精灵(other)发生碰撞
MoveIntoBounds(方法)	call Ball1.MoveIntoBounds	若球移动到画布边界外,可调用本方法将其拉回界内
MoveTo(方法)	call Ball1.MoveTo x y	把球体移动到指定的位置(x,y)
PointInDirection(方法)	call Ball1.PointInDirection x y	将球旋转指向坐标(x,y)
PointTowards(方法)	call Ball1.PointTowards target	让球向目标精灵(target)移动

续表

模块名称	模块及模块说明
Enabled(属性)	set Ball1.Enabled to / Ball1.Enabled 球体的可用性
Heading(属性)	set Ball1.Heading to / Ball1.Heading 球体的方向
Interval(属性)	set Ball1.Interval to / Ball1.Interval 球体的移动频率,单位为毫秒
PaintColor(属性)	set Ball1.PaintColor to / Ball1.PaintColor 球体的颜色
Radius(属性)	set Ball1.Radius to / Ball1.Radius 球体的半径
Visible(属性)	set Ball1.Visible to / Ball1.Visible 球体的可见性
Speed(属性)	set Ball1.Speed to / Ball1.Speed 球体的移动速度,单位为像素
X(属性)	set Ball1.X to / Ball1.X 球体位置的 X 坐标
Y(属性)	set Ball1.Y to / Ball1.Y 球体位置的 Y 坐标
Z(属性)	set Ball1.Z to / Ball1.Z 球体位置的 Z 坐标
Ball1(实例)	component Ball1 球体的一个实例

2. ImageSprite(图像精灵)

ImageSprite 是一种可在画布中自由移动的图像，并可与球体(Ball)、其他图像精灵和画布边缘产生碰撞效果，因此图像精灵经常用于开发游戏。

附表 B.18　ImageSprite 控件说明

模块名称	模块及模块说明	
CollidedWith(事件)	when ImageSprite1.CollidedWith other do	碰撞事件，图像精灵与其他精灵发生碰撞的事件
Dragged(事件)	when ImageSprite1.Dragged startX startY prevX prevY currentX currentY do	拖曳事件，当手指在画布上拖曳图像精灵时触发本事件。 startX、startY：表示拖曳事件开始时 X、Y 坐标值； prevX、prevY：表示上一个拖曳事件产生时的 X、Y 坐标值； currentX、currentY：表示产生该事件时的 X、Y 坐标轴数值
Flung(事件)	when ImageSprite1.Flung x y speed heading xvel yvel do	滑动事件，只有手指在画布上快速滑动时触发本事件。 x、y：滑动事件初始坐标； speed：滑动事件的速度(单位为像素/毫秒)； heading：滑动事件的角度(0～360)； xvel、yvel：速度在 X 轴和 Y 轴的分量
EdgeReached(事件)	when ImageSprite1.EdgeReached edge do	触壁事件，当画布精灵触碰画布边缘时触发本事件，并返回所到达边缘的位置信息，其中 edge 值是返回的边缘位置信息
NoLonger CollidingWith(事件)	when ImageSprite1.NoLongerCollidingWith other do	不再碰撞事件，当图片精灵与其他精灵从碰撞状态分离时触发本事件
Touched(事件)	when ImageSprite1.Touched x y do	触摸事件，当用户对图片精灵进行触摸时触发本事件。 x、y：触控点的 X、Y 轴坐标值

续表

模块名称	模块及模块说明	
TouchDown(事件)	when ImageSprite1.TouchDown x y do	按下事件,当用户对图片精灵进行按下操作时触发本事件。 x、y:手指按下时触摸点的X、Y轴坐标值
TouchUp(事件)	when ImageSprite1.TouchUp x y do	抬起事件,当用户对图片精灵进行按下操作后抬起时触发本事件。 x、y:手指抬起时触摸点的X、Y轴坐标值
Bounce(方法)	call ImageSprite1.Bounce edge	图片精灵反弹,一般在图片精灵与精灵碰撞后使用
CollidingWith(方法)	call ImageSprite1.CollidingWith other	返回图片精灵是否与指定精灵(other)发生碰撞
MoveIntoBounds(方法)	call ImageSprite1.MoveIntoBounds	若图片精灵移动到画布边界外,可调用本方法将其拉回界内
MoveTo(方法)	call ImageSprite1.MoveTo x y	把图片精灵移动到指定的位置(x,y)
PointInDirection(方法)	call ImageSprite1.PointInDirection x y	将图片精灵旋转指向坐标(x,y)
PointTowards(方法)	call ImageSprite1.PointTowards target	让图片精灵向指定的目标精灵(target)移动
Enabled(属性)	set ImageSprite1.Enabled to ImageSprite1.Enabled 图片精灵的可用性	
Heading(属性)	set ImageSprite1.Heading to ImageSprite1.Heading 图片精灵的方向	
Interval(属性)	set ImageSprite1.Interval to ImageSprite1.Interval 图片精灵的移动频率,单位为毫秒	

续表

模块名称	模块及模块说明
Picture(属性)	set ImageSprite1.Picture to ImageSprite1.Picture 图片精灵的背景图片
Rotates(属性)	set ImageSprite1.Rotates to ImageSprite1.Rotates 图片精灵是否已经旋转到 Heading 指定的方向。在用户更改 Heading 后,返回 true 表示已经旋转完毕,返回 false 表示还没有完成旋转
Visible(属性)	set ImageSprite1.Visible to ImageSprite1.Visible 图片精灵的可见性
Speed(属性)	set ImageSprite1.Speed to ImageSprite1.Speed 图片精灵移动的速度,单位为像素
Height(属性)	set ImageSprite1.Height to ImageSprite1.Height 图片精灵的高度
Width(属性)	set ImageSprite1.Width to ImageSprite1.Width 图片精灵的宽度
X(属性)	set ImageSprite1.X to ImageSprite1.X 图片精灵的 X 坐标
Y(属性)	set ImageSprite1.Y to ImageSprite1.Y 图片精灵的 Y 坐标
Z(属性)	set ImageSprite1.Z to ImageSprite1.Z 图片精灵的 Z 坐标
ImageSprite1(实例)	component ImageSprite1 图片精灵控件的一个实例

B.4 Social(社交)控件

社交控件共 6 个，包括 ContactPicker、EmailPicker、PhoneCall、PhoneNumberPicker、Texting 和 Twitter，各个控件的事件、属性和方法如附表 B.19～附表 B.24 所示。

1. ContactPicker(选取联系人)

ContactPicker 的功能是让用户从手机的通讯录中获得联系人信息，这些信息包括联系人的姓名、头像和电子邮件地址。

附表 B.19 ContactPicker 控件说明

模块名称	模块及模块说明	
AfterPicking(事件)	when ContactPicker1.AfterPicking do	选择后事件，在用户选择目标联系人后产生
BeforePicking(事件)	when ContactPicker1.BeforePicking do	选择前事件，在用户打开通讯录，但是尚未选择目标联系人时产生
GotFocus(事件)	when ContactPicker1.GotFocus do	获取焦点事件
LostFocus(事件)	when ContactPicker1.LostFocus do	失去焦点事件
Open(方法)	call ContactPicker1.Open	打开联系人列表供选择
BackgroundColor(属性)	set ContactPicker1.BackgroundColor to ContactPicker1.BackgroundColor 背景颜色	
ContactName(属性)	ContactPicker1.ContactName 联系人姓名	
EmailAddress(属性)	ContactPicker1.EmailAddress 联系人地址	

续表

模块名称	模块及模块说明
Picture(属性)	ContactPicker1.Picture 联系人头像
Enabled(属性)	set ContactPicker1.Enabled to ContactPicker1.Enabled 控件的可用性
Image(属性)	set ContactPicker1.Image to ContactPicker1.Image 控件的背景图片
Text(属性)	set ContactPicker1.Text to ContactPicker1.Text 控件显示的文本
TextColor(属性)	set ContactPicker1.TextColor to ContactPicker1.TextColor 控件的文本颜色
Visible(属性)	set ContactPicker1.Visible to ContactPicker1.Visible 控件的可见性
Height(属性)	set ContactPicker1.Height to ContactPicker1.Height 控件的高度
Width(属性)	set ContactPicker1.Width to ContactPicker1.Width 控件的宽度
ContactPicker1(实例)	component ContactPicker1 选取联系人的一个实例

2. EmailPicker(邮件地址工具)

EmailPicker在用户输入联系人的电子邮件地址时,提供自动完成邮件地址输入的功能。

附表 B.20 EmailPicker 控件说明

模块名称	模块及模块说明
GotFocus(事件)	when EmailPicker1.GotFocus do 获取焦点事件
LostFocus(事件)	when EmailPicker1.LostFocus do 失去焦点事件
BackgroundColor(属性)	set EmailPicker1.BackgroundColor to EmailPicker1.BackgroundColor 控件的背景颜色
Enabled(属性)	set EmailPicker1.Enabled to EmailPicker1.Enabled 控件的可用性
FontSize(属性)	set EmailPicker1.FontSize to EmailPicker1.FontSize 控件的字体大小
Hint(属性)	set EmailPicker1.Hint to EmailPicker1.Hint 控件的提示内容
Text(属性)	set EmailPicker1.Text to EmailPicker1.Text 控件的显示文本内容
TextColor(属性)	set EmailPicker1.TextColor to EmailPicker1.TextColor 控件的显示文本颜色
Visible(属性)	set EmailPicker1.Visible to EmailPicker1.Visible 控件的可见性
Height(属性)	set EmailPicker1.Height to EmailPicker1.Height 控件的高度

续表

模 块 名 称	模块及模块说明
Width(属性)	set EmailPicker1.Width to / EmailPicker1.Width 控件的宽度
EmailPicker1(实例)	component EmailPicker1 邮件地址工具的一个实例

3. PhoneCall(拨号)

PhoneCall 是一个非可视化控件,用于向指定的电话号码拨打电话。

附表 B. 21　PhoneCall 控件说明

模 块 名 称	模块及模块说明
MakePhoneCall(方法)	call PhoneCall1.MakePhoneCall　调用手机的拨号界面,拨打PhoneNumber 属性中的电话号码
PhoneNumber(属性)	set PhoneCall1.PhoneNumber to / PhoneCall1.PhoneNumber 拨号的电话号码
PhoneCall1(实例)	component PhoneCall1 拨号控件的一个实例

4. PhoneNumberPicker(选取号码)

PhoneNumberPicker 可以获取手机通讯录中的联系人信息,这些信息包括联系人的姓名、头像、电子邮件地址和电话号码。

附表 B. 22　PhoneNumberPicker 控件说明

模块名称	模块及模块说明
AfterPicking(事件)	when PhoneNumberPicker1.AfterPicking do　选择后事件,在用户选择目标联系人后产生
BeforePicking(事件)	when PhoneNumberPicker1.BeforePicking do　选择前事件,在用户打开通讯录,但是尚未选择目标联系人时产生

续表

模块名称	模块及模块说明
GotFocus(事件)	when PhoneNumberPicker1.GotFocus do 获取焦点事件
LostFocus(事件)	when PhoneNumberPicker1.LostFocus do 失去焦点事件
Open(方法)	call PhoneNumberPicker1.Open 打开联系人列表供选择
BackgroundColor(属性)	set PhoneNumberPicker1.BackgroundColor to PhoneNumberPicker1.BackgroundColor 控件的背景颜色
ContactName(属性)	PhoneNumberPicker1.ContactName 目标联系人姓名
EmailAddress(属性)	PhoneNumberPicker1.EmailAddress 目标联系人电子邮件地址
Picture(属性)	PhoneNumberPicker1.Picture 目标联系人头像
PhoneNumber(属性)	PhoneNumberPicker1.PhoneNumber 目标联系人电话号码
Enabled(属性)	set PhoneNumberPicker1.Enabled to PhoneNumberPicker1.Enabled 控件的可用性
Image(属性)	set PhoneNumberPicker1.Image to PhoneNumberPicker1.Image 控件的背景图片
Text(属性)	set PhoneNumberPicker1.Text to PhoneNumberPicker1.Text 控件的显示文本信息

续表

模块名称	模块及模块说明
TextColor(属性)	set PhoneNumberPicker1.TextColor to PhoneNumberPicker1.TextColor 控件的文本颜色
Visible(属性)	set PhoneNumberPicker1.Visible to PhoneNumberPicker1.Visible 控件的可见性
Height(属性)	set PhoneNumberPicker1.Height to PhoneNumberPicker1.Height 控件的高度
Width(属性)	set PhoneNumberPicker1.Width to PhoneNumberPicker1.Width 控件的宽度
PhoneNumberPicker1(实例)	component PhoneNumberPicker1 选取号码控件的一个实例

5. Texting(短信息)

Texting是非可视化控件,主要用来发送和接收短信息。

附表B.23 Texting控件说明

模块名称	模块及模块说明	
MessageReceived(事件)	when Texting1.MessageReceived number messageText do	信息接收事件,在接收短信后产生,number是发送方的电话号码,messageText是短信息内容
SendMessage(方法)	call Texting1.SendMessage	发送短信
GoogleVoiceEnabled(属性)	set Texting1.GoogleVoiceEnabled to Texting1.GoogleVoiceEnabled 是否允许使用GoogleVoice服务	

续表

模块名称	模块及模块说明
Message(属性)	set Texting1.Message to Texting1.Message 短信的内容
PhoneNumber(属性)	set Texting1.PhoneNumber to Texting1.PhoneNumber 发短信的目标电话号码
ReceivingEnabled(属性)	set Texting1.ReceivingEnabled to Texting1.ReceivingEnabled 是否允许接收短信
Texting1(实例)	component Texting1 短信的一个实例

6. Twitter(推特)

Twitter 是一个社交网络和一个微博客服务,可以让用户更新不超过 140 个字符的消息,通过该模块可以调用 Twitter 的服务。

附表 B. 24 Twitter 控件说明

模块名称	模块及模块说明	
DirectMessageReceived(事件)	when Twitter1.DirectMessagesReceived messages do	通过 RequestDirectMessages 方法,获取到所查询的信息的事件
FollowersReceived(事件)	when Twitter1.FollowersReceived followers2 do	通过 RequestFollowers 方法获取到所查询的在线好友名单的事件
FriendTimelineReceived(事件)	when Twitter1.FriendTimelineReceived timeline do	通过 RequestFriendTime 方法获取到所查询的信息所产生的事件
IsAuthorized(事件)	when Twitter1.IsAuthorized do	用户登录验证通过或验证该用户已存在的事件

续表

模块名称	模块及模块说明	
MentionsReceived(事件)	when Twitter1.MentionsReceived mentions do	登录用户通过调用 RequestMentions 后，获取结果时产生的事件
SearchSuccessful(事件)	when Twitter1.SearchSuccessful searchResults do	Twitter 中搜索成功事件
Authorize(方法)	call Twitter1.Authorize	给用户呈现一个 Twitter 的登录页面
CheckAuthorized(方法)	call Twitter1.CheckAuthorized	检测用户输入的用户名及密码信息是否正确
DeAuthorize(方法)	call Twitter1.DeAuthorize	退出已登录的 Twitter 应用程序
DirectMessage(方法)	call Twitter1.DirectMessage user message	向用户(user)发送消息(message)
Follow(方法)	call Twitter1.Follow user	关注指定的使用者(user)
RequestDirectMessages(方法)	call Twitter1.RequestDirectMessages	接收最新的消息
RequestFollowers(方法)	call Twitter1.RequestFollowers	获取正在关注我的用户列表
RequestFriendTimeline(方法)	call Twitter1.RequestFriendTimeline	获取关注我的用户的最新消息
RequestMentions(方法)	call Twitter1.RequestMentions	获取有关其他人提及登录用户的最新消息列表
SearchTwitter(方法)	call Twitter1.SearchTwitter query	在 Twitter 中搜索内容
SetStatus(方法)	call Twitter1.SetStatus status	设置用户状态

续表

模块名称	模块及模块说明
StopFollowing(方法)	call Twitter1.StopFollowing user 取消对指定用户(user)的关注
ConsumerKey(属性)	set Twitter1.ConsumerKey to　Twitter1.ConsumerKey Twitter 用来确认用户身份的字符串,该字符串可以从 twitter.com/oauth_clients/new 中获取
ConsumerSecret(属性)	set Twitter1.ConsumerSecret to　Twitter1.ConsumerSecret Twitter 用来确认用户身份的字符串
DirectMessages(属性)	Twitter1.DirectMessages 发送的消息文本
Followers(属性)	Twitter1.Followers 关注用户中在线用户的列表
FriendTimeline(属性)	Twitter1.FriendTimeline 关注用户的最新消息
Mentions(属性)	Twitter1.Mentions 提及登录者的最新消息列表
SearchResults(属性)	Twitter1.SearchResults SearchTwitter 方法的搜索结果
Username(属性)	Twitter1.Username 获取登录用户的用户名
Twitter1(实例)	component Twitter1 Twitter 的一个实例

B.5 Sensors(传感器)模块

感应器控件共三个，包括 AccelerometerSensor、LocationSensor 和 OrientationSensor，各个控件的事件、属性和方法如附表 B.25～附表 B.27 所示。

1. AccelerometerSensor(加速传感器)

AccelerometerSensor 的主要功能是获取手机加速度感应器的状态，并侦测设备三维空间的晃动情况。

附表 B.25 AccelerometerSensor 控件说明

模块名称	模块及模块说明
AccelerationChanged(事件)	when AccelerometerSensor1.AccelerationChanged xAccel yAccel zAccel do 加速度感应器的值改变事件
Shaking(事件)	when AccelerometerSensor1.Shaking do 手机晃动事件
Available(属性)	AccelerometerSensor1.Available 手机是否存在方向加速度传感器
Enabled(属性)	set AccelerometerSensor1.Enabled to AccelerometerSensor1.Enabled 控件的可用性
MinimumInterval(属性)	set AccelerometerSensor1.MinimumInterval to AccelerometerSensor1.MinimumInterval 手机振动的最小时间间隔
XAccel(属性)	AccelerometerSensor1.XAccel 加速度传感器 X 轴的变化量
YAccel(属性)	AccelerometerSensor1.YAccel 加速度传感器 Y 轴的变化量

续表

模块名称	模块及模块说明
ZAccel(属性)	AccelerometerSensor1.ZAccel 加速度传感器 Z 轴的变化量
AccelerometerSensor1(实例)	component AccelerometerSensor1 加速度传感器的一个实例

2. LocationSensor(位置传感器)

LocationSensor 的主要功能是使用设备的 GPS 或者其他定位方法(移动基站或无线网络)获取手机的当前位置信息。

附表 B. 26 LocationSensor 控件说明

模块名称	模块及模块说明	
LocationChanged(事件)	when LocationSensor1.LocationChanged latitude longitude altitude do	位置改变事件
StatusChanged(事件)	when LocationSensor1.StatusChanged provider status do	设备的服务提供者状态改变事件
LatitudeFromAddress(方法)	call LocationSensor1.LatitudeFromAddress locationName	获取指定地址的纬度
LongitudeFromAddress(方法)	call LocationSensor1.LongitudeFromAddress locationName	获取指定地址的经度
Accuracy(属性)	LocationSensor1.Accuracy	手机所在位置的精度等级
Altitude(属性)	LocationSensor1.Altitude	手机所在位置的海拔高度
AvailableProviders(属性)	LocationSensor1.AvailableProviders	手机可用的服务提供者清单

续表

模块名称	模块及模块说明	
CurrentAddress(属性)	LocationSensor1.CurrentAddress	手机的当前位置
HasAccuracy(属性)	LocationSensor1.HasAccuracy	手机是否可以回传精度
HasAltitude(属性)	LocationSensor1.HasAltitude	手机是否可以回传海拔高度
HasLongitudeLatitude(属性)	LocationSensor1.HasLongitudeLatitude	手机是否可以回传经度和纬度
Latitude(属性)	LocationSensor1.Latitude	手机所在位置的纬度
Longitude(属性)	LocationSensor1.Longitude	手机所在位置的经度
DistanceInterval(属性)	set LocationSensor1.DistanceInterval to LocationSensor1.DistanceInterval 位置更新的最小变化距离	
Enabled(属性)	set LocationSensor1.Enabled to LocationSensor1.Enabled 位置传感器是否可用	
ProviderLocked(属性)	set LocationSensor1.ProviderLocked to LocationSensor1.ProviderLocked Android设备锁定现在的服务提供者	
ProviderName(属性)	set LocationSensor1.ProviderName to LocationSensor1.ProviderName 位置服务的提供者名称	
TimeInterval(属性)	set LocationSensor1.TimeInterval to LocationSensor1.TimeInterval 位置更新的最小时间间隔	
LocationSensor1(实例)	component LocationSensor1 位置传感器的一个实例	

3. OrientationSensor(位置传感器)

OrientationSensor 用来测定手机的方向变化。

附表 B. 27　OrientationSensor 控件说明

模块名称	模块及模块说明	
OrientationChanged(事件)	when OrientationSensor1.OrientationChanged azimuth pitch roll do	方向变化事件
Angle(属性)	OrientationSensor1.Angle	手机的倾斜角大小
Available(属性)	OrientationSensor1.Available	手机是否存在方向传感器
Azimuth(属性)	OrientationSensor1.Azimuth	手机的方位角
Magnitude(属性)	OrientationSensor1.Magnitude	手机的倾斜程度,用 0～1 的数字表示
Pitch(属性)	OrientationSensor1.Pitch	手机的翻转角
Roll(属性)	OrientationSensor1.Roll	手机的转动角
Enabled(属性)	set OrientationSensor1.Enabled to OrientationSensor1.Enabled 方向感应器是否可用	
OrientationSensor1(实例)	component OrientationSensor1 方向传感器的一个实例	

B.6 Screen Arrangement(屏幕布局)控件

屏幕布局控件共三个,包括 HorizontalArrangement、TableArrangement 和 VerticalArrangement,各个控件的事件、属性和方法如附表 B. 28～附表 B. 30 所示。

1. HorizontalArrangement(水平布局)

HorizontalArrangement 可以将多个控件横向排列于其中，但每列仅包含一个界面控件。

附表 B.28　HorizontalArrangement 控件说明

模块名称	模块及模块说明
AlignHorizontal(属性)	set HorizontalArrangement1.AlignHorizontal to　HorizontalArrangement1.AlignHorizontal 控件在布局中水平方向的排布方式，可以左对齐、右对齐、居中或自动
AlignVertical(属性)	set HorizontalArrangement1.AlignVertical to　HorizontalArrangement1.AlignVertical 控件在布局中竖直方向的排列方式，可以上对齐、下对齐、居中或自动
Height(属性)	set HorizontalArrangement1.Height to　HorizontalArrangement1.Height 水平布局的高度
Visible(属性)	set HorizontalArrangement1.Visible to　HorizontalArrangement1.Visible 水平布局是否可见
Width(属性)	set HorizontalArrangement1.Width to　HorizontalArrangement1.Width 水平布局的宽度
HorizontalArrangement1(实例)	component HorizontalArrangement1 水平布局的一个实例

2. VerticalArrangement(垂直布局)

VerticalArrangement 可以竖直排列多个控件，但每行仅包含一个界面控件。

附表 B.29　VerticalArrangement 控件说明

模块名称	模块及模块说明
AlignHorizontal(属性)	set VerticalArrangement1.AlignHorizontal to　VerticalArrangement1.AlignHorizontal 控件在布局中水平方向的排布方式，可以左对齐、右对齐、居中或自动

续表

模块名称	模块及模块说明
AlignVertical（属性）	set VerticalArrangement1.AlignVertical to VerticalArrangement1.AlignVertical 控件在布局中竖直方向的排列方式，可以上对齐、下对齐、居中或自动
Height（属性）	set VerticalArrangement1.Height to VerticalArrangement1.Height 垂直布局的高度
Visible（属性）	set VerticalArrangement1.Visible to VerticalArrangement1.Visible 垂直布局是否可见
Width（属性）	set VerticalArrangement1.Width to VerticalArrangement1.Width 垂直布局的宽度
VerticalArrangement1（实例）	component VerticalArrangement1 垂直布局的一个实例

3. TableArrangement（表格布局）

TableArrangement 可以划分多行多列，每个网格中可以放一个或者多个界面控件。

附表 B.30　TableArrangement 控件说明

模块名称	模块及模块说明
Height（属性）	set TableArrangement1.Height to TableArrangement1.Height 表格布局的高度
Visible（属性）	set TableArrangement1.Visible to TableArrangement1.Visible 表格布局是否可见
Width（属性）	set TableArrangement1.Width to TableArrangement1.Width 表格布局的宽度
TableArrangement1（实例）	component TableArrangement1 表格布局的一个实例

B.7 LEGO MINDSTORMS(乐高机器人)控件

乐高 NXT 智慧型机器人控件共 7 个，包括 NxtColorSensor、NxtDirectCommands、NxtDrive、NxtLightSensor、NxtSoundSensor、NxtTouchSensor 和 NxtUltrasonicSensor，各个控件的事件、属性和方法如附表 B.31～附表 B.37 所示。

1. NxtColorSensor(Nxt 颜色感应器)

NxtColorSensor 模块用来控制乐高 NXT 机器人上的颜色感应器。

附表 B.31 NxtColorSensor 控件说明

模块名称	模块及模块说明	
AboveRange(事件)	when NxtColorSensor1.AboveRange do	当光值高于指定的范围时触发的事件
BelowRange(事件)	when NxtColorSensor1.BelowRange do	当光值低于指定的范围时触发的事件
ColorChanged(事件)	when NxtColorSensor1.ColorChanged color do	检测到颜色发生改变时触发的事件
WithinRange(事件)	when NxtColorSensor1.WithinRange do	当光值在指定的范围内时触发的事件
GetColor(方法)	call NxtColorSensor1.GetColor	返回颜色感应器所检测到的颜色
GetLightLevel(方法)	call NxtColorSensor1.GetLightLevel	返回光值强度(介于 0～1023 的整数)
AboveRange-EventEnabled(属性)	set NxtColorSensor1.AboveRangeEventEnabled to / NxtColorSensor1.AboveRangeEventEnabled 当光值大于 TopOfRange 时是否触发 AboveRange 事件	
BelowRange-EventEnabled(属性)	set NxtColorSensor1.BelowRangeEventEnabled to / NxtColorSensor1.BelowRangeEventEnabled 当光值低于 BottomOfRange 时是否触发 BelowRange 事件	

续表

模块名称	模块及模块说明
BottomOfRange（属性）	set NxtColorSensor1.BottomOfRange to NxtColorSensor1.BottomOfRange 设置触发 WithinRange、BelowRange、AboveRange 等事件的最小值
ColorChanged-EventEnabled（属性）	set NxtColorSensor1.ColorChangedEventEnabled to NxtColorSensor1.ColorChangedEventEnabled 当检测到颜色发生变化时，设定是否触发 ColorChanged 事件
DetectColor（属性）	set NxtColorSensor1.DetectColor to NxtColorSensor1.DetectColor 设定颜色感应器是检测颜色还是光值，设为 true 时检测颜色变化，反之检测光值变化
GenerateColor（属性）	set NxtColorSensor1.GenerateColor to NxtColorSensor1.GenerateColor 设定颜色感应器是否发光
TopOfRange（属性）	set NxtColorSensor1.TopOfRange to NxtColorSensor1.TopOfRange 设置触发 WithinRange、BelowRange、AboveRange 等事件的最大值
WithinRange-EventEnabled（属性）	set NxtColorSensor1.WithinRangeEventEnabled to NxtColorSensor1.WithinRangeEventEnabled 当光值介于 BottomOfRange 与 TopOfRange 之间时，是否触发 WithinRange 事件
NxtColorSensor1（实例）	component NxtColorSensor1 颜色感应器的一个实例

2. NxtDirectCommands（Nxt 通信规范）

NxtDirectCommands 定义了对 NXT 智慧机器人的通信命令，通过该命令集可以直接对机器人进行控制。

附表 B.32　NxtDirectCommands 控件说明

模块名称	模块及模块说明	
DeleteFile（方法）	call NxtDirectCommands1.DeleteFile fileName	删除机器人上的文件
DownloadFile（方法）	call NxtDirectCommands1.DownloadFile source destination	将文件下载到机器人上

续表

模块名称	模块及模块说明	
GetBatteryLevel(方法)	call NxtDirectCommands1.GetBatteryLevel	获得机器人的电池电量
GetBrickName(方法)	call NxtDirectCommands1.GetBrickName	获得 NXT 主机名称
GetCurrentProgram-Name(方法)	call NxtDirectCommands1.GetCurrentProgramName	获得机器人正在运行的程序名称
GetFirmwareVersion(方法)	call NxtDirectCommands1.GetFirmwareVersion	获得机器人固件和通信协议版本号
GetInputValues(方法)	call NxtDirectCommands1.GetInputValues sensorPortLetter	从机器人指定的输入端读取信息
GetOutputState(方法)	call NxtDirectCommands1.GetOutputState motorPortLetter	读取机器人指定输出端的状态
KeepAlive(方法)	call NxtDirectCommands1.KeepAlive	使机器人保持开机的状态
ListFiles(方法)	call NxtDirectCommands1.ListFiles wildcard	以列表的方式返回机器人中符合条件 wildcard 的文件
LsGetStatus(方法)	call NxtDirectCommands1.LsGetStatus sensorPortLetter	获取指定端口的低速通信状态
LsRead(方法)	call NxtDirectCommands1.LsRead sensorPortLetter	从机器人指定输入端读取串行通信的信息
LsWrite(方法)	call NxtDirectCommands1.LsWrite sensorPortLetter list rxDataLength	对机器人指定输入端读取低速信息
MessageRead(方法)	call NxtDirectCommands1.MessageRead mailbox	从机器人的指定信箱读取信息
MessageWrite(方法)	call NxtDirectCommands1.MessageWrite mailbox message	对机器人的指定信箱写入信息
PlaySoundFile(方法)	call NxtDirectCommands1.PlaySoundFile fileName	播放机器人上的指定音效文件

续表

模块名称	模块及模块说明	
PlayTone(方法)	call NxtDirectCommands1.PlayTone frequencyHz durationMs	让机器人发出指定长度和音频的声音
ResetInputScaled-Value(方法)	call NxtDirectCommands1.ResetInputScaledValue sensorPortLetter	重设指定输入端口的标准值
ResetMotorPosition(方法)	call NxtDirectCommands1.ResetMotorPosition motorPortLetter relative	重设电机的位置
SetBrickName(方法)	call NxtDirectCommands1.SetBrickName name	设定 NXT 的主机名称
SetInputMode(方法)	call NxtDirectCommands1.SetInputMode sensorPortLetter sensorType sensorMode	设定机器人指定输入端的状态，sensorPortLetter 为输入端编号，sensorType 为感应器类型，sensorMode 为感应器返回值的格式
SetOutputState(方法)	call NxtDirectCommands1.SetOutputState motorPortLetter power mode regulationMode turnRatio runState tachoLimit	设定机器人的指定输出端的状态
StartProgram(方法)	call NxtDirectCommands1.StartProgram programName	运行已下载到机器上的程序
StopProgram(方法)	call NxtDirectCommands1.StopProgram	停止机器人正在运行的程序
StopSoundPlayback(方法)	call NxtDirectCommands1.StopSoundPlayback	停止播放声音
NxtDirectCommands1(实例)	component NxtDirectCommands1	通信规格的一个实例

3. NxtDrive(Nxt 电机)

NxtDrive 模块用来控制 NXT 机器人的马达,进而控制机器人的前进、后退、拐弯等动作。

附表 B.33 NxtDrive 控件说明

模块名称	模块及模块说明	
MoveBackward(方法)	call NxtDrive1.MoveBackward power distance	机器人以指定的电力(power)后退指定的距离(distance)
MoveBackwardIndefinitely(方法)	call NxtDrive1.MoveBackwardIndefinitely power	让机器人以指定的电力(power)持续后退,power 范围为－100～100
MoveForward(方法)	call NxtDrive1.MoveForward power distance	机器人以指定的电力(power)前进指定的距离(distance)
MoveForwardIndefinitely(方法)	call NxtDrive1.MoveForwardIndefinitely power	让机器人以指定的电力(power)持续地前进,power 范围为－100～100
Stop(方法)	call NxtDrive1.Stop	所有的电机停止转动
TurnClockwiseIndefinitely(方法)	call NxtDrive1.TurnClockwiseIndefinitely power	让机器人以指定的电力(power)持续地顺时针转动,power 范围为－100～100
TurnCounterClockwise-Indefinitely(方法)	call NxtDrive1.TurnCounterClockwiseIndefinitely power	让机器人以指定的电力(power)持续地逆时针转动,power 范围为－100～100

续表

模块名称	模块及模块说明
StopBeforeDisconnect（属性）	set NxtDrive1.StopBeforeDisconnect to NxtDrive1.StopBeforeDisconnect 设定是否在断开连接之前将电机停止运转
NxtDrive1（实例）	component NxtDrive1 控制电机的一个实例

4. NxtLightSensor（Nxt 光感应器）

NxtLightSensor 模块用来控制乐高 NXT 机器人上的光感应器。

附表 B.34　NxtLightSensor 控件说明

模块名称	模块及模块说明
AboveRange（事件）	when NxtLightSensor1.AboveRange do 当光值高于指定的范围时触发的事件
BelowRange（事件）	when NxtLightSensor1.BelowRange do 当光值低于指定的范围时触发的事件
WithinRange（事件）	when NxtLightSensor1.WithinRange do 当光值在指定的范围内时触发的事件
GetLightLevel（方法）	call NxtLightSensor1.GetLightLevel 返回光值强度，强度值介于 0～1023 之间
AboveRange-EventEnabled（属性）	set NxtLightSensor1.AboveRangeEventEnabled to NxtLightSensor1.AboveRangeEventEnabled 设置当光值大于 TopOfRange 时，是否触发 AboveRange 事件
BelowRange-EventEnabled（属性）	set NxtLightSensor1.BelowRangeEventEnabled to NxtLightSensor1.BelowRangeEventEnabled 设置当光值低于 BottomOfRange 时，是否触发 BelowRange 事件
BottomOf-Range（属性）	set NxtLightSensor1.BottomOfRange to NxtLightSensor1.BottomOfRange 设置触发 WithinRange、BelowRange、AboveRange 等事件的最小值

续表

模块名称	模块及模块说明
GenerateLight（属性）	set NxtLightSensor1.GenerateLight to NxtLightSensor1.GenerateLight 光感应器的前端是否会发光
TopOfRange（属性）	set NxtLightSensor1.TopOfRange to NxtLightSensor1.TopOfRange 设置触发 WithinRange、BelowRange、AboveRange 等事件的最大值
WithinRange-EventEnabled（属性）	set NxtLightSensor1.WithinRangeEventEnabled to NxtLightSensor1.WithinRangeEventEnabled 设置当光值介于 BottomOfRange 与 TopOfRange 之间时，是否触发 WithinRange 事件
NxtLight-Sensor1（实例）	component NxtLightSensor1 光感应器的一个实例

5. NxtSoundSensor（Nxt 声音感应器）

NxtSoundSensor 模块用来控制乐高 NXT 机器人上的声音感应器。

附表 B.35　NxtSoundSensor 控件说明

模块名称	模块及模块说明	
AboveRange（事件）	when NxtSoundSensor1.AboveRange do	当音量高于指定的范围时触发的事件
BelowRange（事件）	when NxtSoundSensor1.BelowRange do	当音量低于指定的范围时触发的事件
WithinRange（事件）	when NxtSoundSensor1.WithinRange do	当音量介于指定的范围内时触发的事件
GetSoundLevel（方法）	call NxtSoundSensor1.GetSoundLevel	返回音量强度，强度值介于 0～1023 之间
AboveRange-EventEnabled（属性）	set NxtSoundSensor1.AboveRangeEventEnabled to NxtSoundSensor1.AboveRangeEventEnabled 设置当音量超过 TopOfRange 时，是否触发 AboveRange 事件	

续表

模块名称	模块及模块说明
BelowRange-EventEnabled（属性）	set NxtSoundSensor1.BelowRangeEventEnabled to / NxtSoundSensor1.BelowRangeEventEnabled 设置当音量低于 BottomOfRange 时，是否触发 BelowRange 事件
BottomOfRange（属性）	set NxtSoundSensor1.BottomOfRange to / NxtSoundSensor1.BottomOfRange 设置触发 WithinRange、BelowRange、AboveRange 等事件的最小值
TopOfRange（属性）	set NxtSoundSensor1.TopOfRange to / NxtSoundSensor1.TopOfRange 设置触发 WithinRange、BelowRange、AboveRange 等事件的最大值
WithinRange-EventEnabled（属性）	set NxtSoundSensor1.WithinRangeEventEnabled to / NxtSoundSensor1.WithinRangeEventEnabled 设置当音量介于 BottomOfRange 与 TopOfRange 之间时，是否触发 WithinRange 事件
NxtSoundSensor1（实例）	component NxtSoundSensor1 声音感应器的一个实例

6. NxtTouchSensor（Nxt 触摸感应器）

NxtTouchSensor 模块用来控制乐高 NXT 机器人上的触摸感应器。

附表 B. 36 NxtTouchSensor 控件说明

模块名称	模块及模块说明	
Pressed（事件）	when NxtTouchSensor1.Pressed do	当触摸感应器被按下时触发的事件
Released（事件）	when NxtTouchSensor1.Released do	当触摸感应器被放开时触发的事件
IsPressed（方法）	call NxtTouchSensor1.IsPressed	返回触摸感应器是否被按下
PressedEventEnabled（属性）	set NxtTouchSensor1.PressedEventEnabled to / NxtTouchSensor1.PressedEventEnabled 设置当触摸感应器被按下时是否能够触发 Pressed 事件	

续表

模块名称	模块及模块说明
ReleasedEvent-Enabled(属性)	set NxtTouchSensor1.ReleasedEventEnabled to NxtTouchSensor1.ReleasedEventEnabled 设置放开触摸感应器时是否能够触发 Released 事件
NxtTouchSensor1(实例)	component NxtTouchSensor1 触摸感应器的一个实例

7. NxtUltrasonicSensor(Nxt 超音波感应器)

NxtUltrasonicSensor 模块的主要功能是用来控制乐高 NXT 机器人上的超音波感应器,如附表 B.37 所示。

附表 B.37　NxtUltrasonicSensor 控件说明

模块名称	模块及模块说明	
AboveRange(事件)	when NxtUltrasonicSensor1.AboveRange do	当距离大于指定的范围时触发的事件
BelowRange(事件)	when NxtUltrasonicSensor1.BelowRange do	当距离小于指定的范围时触发的事件
WithinRange(事件)	when NxtUltrasonicSensor1.WithinRange do	当距离介于指定的范围之间时触发的事件
GetDistance(方法)	call NxtUltrasonicSensor1.GetDistance	返回距离的大小,单位为厘米,距离值介于 0～254
AboveRange-EventEnabled(属性)	set NxtUltrasonicSensor1.AboveRangeEventEnabled to NxtUltrasonicSensor1.AboveRangeEventEnabled 设置当距离超过 TopOfRange 时,是否触发 AboveRange 事件	
BelowRange-EventEnabled(属性)	set NxtUltrasonicSensor1.BelowRangeEventEnabled to NxtUltrasonicSensor1.BelowRangeEventEnabled 设置当距离小于 BottomOfRange 时,是否触发 BelowRange 事件	

续表

模 块 名 称	模块及模块说明
BottomOfRange（属性）	set NxtUltrasonicSensor1.BottomOfRange to NxtUltrasonicSensor1.BottomOfRange 设置触发 WithinRange、BelowRange、AboveRange 等事件的最小值
TopOfRange（属性）	set NxtUltrasonicSensor1.TopOfRange to NxtUltrasonicSensor1.TopOfRange 设置触发 WithinRange、BelowRange、AboveRange 等事件的最大值
WithinRange-EventEnabled（属性）	set NxtUltrasonicSensor1.WithinRangeEventEnabled to NxtUltrasonicSensor1.WithinRangeEventEnabled 设置当距离介于 BottomOfRange 与 TopOfRange 之间时，是否触发 WithinRange 事件
NxtUltrasonic-Sensor1（实例）	component NxtUltrasonicSensor1 超音波感应器的一个实例

B.8 Other stuff(其他)控件

其他控件共 9 个，包括 ActivityStarter、BarcodeScanner、BluetoothClient、BluetoothServer、Notifice、SpeechRecognizer、TextToSpeech、TinyWebDB 和 Web，各个控件的事件、属性和方法如附表 B. 38～附表 B. 46 所示。

1. ActivityStarter（程序启动器）

ActivityStarter 组件是一个不可见控件，通过该控件可以调用第三方应用程序。

附表 B. 38 ActivityStarter 控件说明

模 块 名 称	模块及模块说明	
AfterActivity（事件）	when ActivityStarter1.AfterActivity result do	调用第三方应用程序完成后触发本事件
ResolveActivity（方法）	call ActivityStarter1.ResolveActivity	返回被调用的第三方应用程序的名称，若未找到则返回空字符串。通过本方法可以确认第三方应用程序是否已安装在手机中
StartActivity（方法）	call ActivityStarter1.StartActivity	启动第三方应用程序

续表

模块名称	模块及模块说明
Action(属性)	set ActivityStarter1.Action to ActivityStarter1.Action 调用第三方应用程序的执行动作
ActivityClass(属性)	set ActivityStarter1.ActivityClass to ActivityStarter1.ActivityClass 调用第三方应用程序的类名称
ActivityPackage(属性)	set ActivityStarter1.ActivityPackage to ActivityStarter1.ActivityPackage 调用第三方应用程序的包名称
DataType(属性)	set ActivityStarter1.DataType to ActivityStarter1.DataType 调用第三方应用程序的数据类型
DataUri(属性)	set ActivityStarter1.DataUri to ActivityStarter1.DataUri 调用第三方应用程序的数据统一资源定位符
ExtraKey(属性)	set ActivityStarter1.ExtraKey to ActivityStarter1.ExtraKey 传递给第三方应用程序的键名称
ExtraValue(属性)	set ActivityStarter1.ExtraValue to ActivityStarter1.ExtraValue 传递给第三方应用程序的键值
ResultName(属性)	set ActivityStarter1.ResultName to ActivityStarter1.ResultName 结果的名称
Result(属性)	ActivityStarter1.Result 结果的内容
ResultType(属性)	ActivityStarter1.ResultType 数据类型

续表

模块名称	模块及模块说明
ResultUri(属性)	ActivityStarter1.ResultUri 统一资源定位符
ActivityStarter1(实例)	component ActivityStarter1 程序启动器的一个实例

2. BarcodeScanner(条码扫描器)

BarcodeScanner 通过启动手机的摄像头来读取条形码或二维码(QR 码)。

附表 B.39 BarcodeScanner 控件说明

模块名称	模块及模块说明	
AfterScan(事件)	when BarcodeScanner1.AfterScan result do	扫描结束事件
DoScan(方法)	call BarcodeScanner1.DoScan	对条形码或二维码进行扫描
Result(属性)	BarcodeScanner1.Result 扫描成功后返回的字符串结果	
BarcodeScanner1(实例)	component BarcodeScanner1 条码扫描器的一个实例	

3. BluetoothClient(蓝牙客户端)

BluetoothClient 是蓝牙通信的客户端,可以调用手机上的蓝牙设备,实现手机与其他蓝牙设备之间的连接。

附表 B.40 BluetoothClient 控件说明

模块名称	模块及模块说明	
BytesAvailableToReceive(方法)	call BluetoothClient1.BytesAvailableToReceive	在不阻塞的情况下,预计可接收的字节数

续表

模块名称	模块及模块说明	
Connect(方法)	call BluetoothClient1.Connect address	通过address与另一个蓝牙设备建立连接,成功则返回true
ConnectWithUUID(方法)	call BluetoothClient1.ConnectWithUUID address uuid	通过address和UUID与另一个蓝牙设备建立连接,成功则返回true
Disconnect(方法)	call BluetoothClient1.Disconnect	中断蓝牙连接
IsDevicePaired(方法)	call BluetoothClient1.IsDevicePaired address	检测是否与指定的设备完成配对
ReceiveSigned1ByteNumber(方法)	call BluetoothClient1.ReceiveSigned1ByteNumber	从连接的蓝牙设备接收一个字节长度的有符号数
ReceiveSigned2ByteNumber(方法)	call BluetoothClient1.ReceiveSigned2ByteNumber	从连接的蓝牙设备接收两个字节长度的有符号数
ReceiveSigned4ByteNumber(方法)	call BluetoothClient1.ReceiveSigned4ByteNumber	从连接的蓝牙设备接收4个字节长度的有符号数
ReceiveSignedBytes(方法)	call BluetoothClient1.ReceiveSignedBytes numberOfBytes	从连接的蓝牙设备接收多个字节的有符号数的值,直到遇到结束符
ReceiveText(方法)	call BluetoothClient1.ReceiveText numberOfBytes	从连接的蓝牙设备接收一个字符串,直到遇到结束符
ReceiveUnsigned1ByteNumber(方法)	call BluetoothClient1.ReceiveUnsigned1ByteNumber	从连接的蓝牙设备接收一个字节长度的无符号数
ReceiveUnsigned2ByteNumber(方法)	call BluetoothClient1.ReceiveUnsigned2ByteNumber	从连接的蓝牙设备接收两个字节长度的无符号数
ReceiveUnsigned4ByteNumber(方法)	call BluetoothClient1.ReceiveUnsigned4ByteNumber	从连接的蓝牙设备接收4个字节长度的无符号数
ReceiveUnsignedBytes(方法)	call BluetoothClient1.ReceiveUnsignedBytes numberOfBytes	从连接的蓝牙设备接收多个字节的无符号数的值,直到遇到结束符

续表

模块名称	模块及模块说明	
Send1ByteNumber（方法）	call BluetoothClient1.Send1ByteNumber number	向连接的蓝牙设备发送一个字节长度的数
Send2ByteNumber（方法）	call BluetoothClient1.Send2ByteNumber number	向连接的蓝牙设备发送两个字节长度的数
Send4ByteNumber（方法）	call BluetoothClient1.Send4ByteNumber number	向连接的蓝牙设备发送4个字节长度的数
SendBytes（方法）	call BluetoothClient1.SendBytes list	向连接的蓝牙设备发送数组
SendText（方法）	call BluetoothClient1.SendText text	向连接的蓝牙设备发送字符串
AddressesAndNames（属性）	BluetoothClient1.AddressesAndNames 已配对蓝牙设备的地址和名称	
Available（属性）	BluetoothClient1.Available Android 设备上的蓝牙可用性	
CharacterEncoding（属性）	set BluetoothClient1.CharacterEncoding to BluetoothClient1.CharacterEncoding 接收信息的字符编码方式	
DelimiterByte（属性）	set BluetoothClient1.DelimiterByte to BluetoothClient1.DelimiterByte 使用 ReceiveText、ReceiveSignedBytes、ReceiveUnsignedBytes 等方法时的结束符	
Enabled（属性）	BluetoothClient1.Enabled 蓝牙客户端是否可用	
HighByteFirst（属性）	set BluetoothClient1.HighByteFirst to BluetoothClient1.HighByteFirst 是否采用高位优先传递的传输方式	
Secure（属性）	set BluetoothClient1.Secure to BluetoothClient1.Secure 是否采用简易安全配对机制	

续表

模 块 名 称	模块及模块说明
IsConnected(属性)	BluetoothClient1.IsConnected 是否已建立连接
BluetoothClient1（实例）	component BluetoothClient1 蓝牙客户端的一个实例

4. BluetoothServer(蓝牙服务端)

BluetoothServer 是蓝牙通信的服务端，可以调用 Android 手机上的蓝牙设备，实现手机与其他蓝牙设备之间的连接。

附表 B. 41　BluetoothServer 控件说明

模 块 名 称	模块及模块说明	
ConnectionAccepted(事件)	when BluetoothServer1.ConnectionAccepted do	蓝牙连接被接受事件
AcceptConnection(方法)	call BluetoothServer1.AcceptConnection serviceName	接受外部蓝牙连接
AcceptConnection-WithUUID(方法)	call BluetoothServer1.AcceptConnectionWithUUID serviceName uuid	接受指定设备的蓝牙连接请求
BytesAvailableTo-Receive(方法)	call BluetoothServer1.BytesAvailableToReceive	在不阻塞的情况下可接收的字节数(估计值)
Disconnect(方法)	call BluetoothServer1.Disconnect	中断蓝牙连接
ReceiveSigned 1ByteNumber(方法)	call BluetoothServer1.ReceiveSigned1ByteNumber	从连接的蓝牙设备接收一个字节长度的有符号数
ReceiveSigned 2ByteNumber(方法)	call BluetoothServer1.ReceiveSigned2ByteNumber	从连接的蓝牙设备接收两个字节长度的有符号数

续表

模块名称	模块及模块说明	
ReceiveSigned4ByteNumber(方法)	call BluetoothServer1.ReceiveSigned4ByteNumber	从连接的蓝牙设备接收 4 个字节长度的有符号数
ReceiveSignedBytes(方法)	call BluetoothServer1.ReceiveSignedBytes numberOfBytes	从连接的蓝牙设备接收多个字节的有符号数的值，直到遇到结束符
ReceiveText(方法)	call BluetoothServer1.ReceiveText numberOfBytes	从连接的蓝牙设备接收一个字符串，直到遇到结束符
ReceiveUnsigned1ByteNumber(方法)	call BluetoothServer1.ReceiveUnsigned1ByteNumber	从连接的蓝牙设备接收一个字节长度的无符号数
ReceiveUnsigned2ByteNumber(方法)	call BluetoothServer1.ReceiveUnsigned2ByteNumber	从连接的蓝牙设备接收两个字节长度的无符号数
ReceiveUnsigned4ByteNumber(方法)	call BluetoothServer1.ReceiveUnsigned4ByteNumber	从连接的蓝牙设备接收 4 个字节长度的无符号数
ReceiveUnsigned-Bytes(方法)	call BluetoothServer1.ReceiveUnsignedBytes numberOfBytes	从连接的蓝牙设备接收多个字节的无符号数的值，直到遇到结束符
Send1ByteNumber(方法)	call BluetoothServer1.Send1ByteNumber number	向连接的蓝牙设备发送一个字节长度的数
Send2ByteNumber(方法)	call BluetoothServer1.Send2ByteNumber number	向连接的蓝牙设备发送两个字节长度的数
Send4ByteNumber(方法)	call BluetoothServer1.Send4ByteNumber number	向连接的蓝牙设备发送 4 个字节长度的数
SendBytes(方法)	call BluetoothServer1.SendBytes list	向连接的蓝牙设备发送数组
SendText(方法)	call BluetoothServer1.SendText text	向连接的蓝牙设备发送字符串

续表

模块名称	模块及模块说明
StopAccepting(方法)	call BluetoothServer1.StopAccepting 不再接受外部连接请求
IsAccepting(属性)	BluetoothServer1.IsAccepting 是否允许蓝牙客户端的连接请求
Available(属性)	BluetoothServer1.Available 手机上的蓝牙可用性
CharacterEncoding(属性)	set BluetoothServer1.CharacterEncoding to BluetoothServer1.CharacterEncoding 接收信息的字符编码方式
DelimiterByte(属性)	set BluetoothServer1.DelimiterByte to BluetoothServer1.DelimiterByte 使用 ReceiveText、ReceiveSignedBytes、ReceiveUnsignedBytes 等方法时的结束符
Enabled(属性)	BluetoothServer1.Enabled 蓝牙服务端是否可用
HighByteFirst(属性)	set BluetoothServer1.HighByteFirst to BluetoothServer1.HighByteFirst 是否采用高位优先传递的传输方式
Secure(属性)	set BluetoothServer1.Secure to BluetoothServer1.Secure 是否采用简易安全配对机制
IsConnected(属性)	BluetoothServer1.IsConnected 是否已建立连接
BluetoothServer1(实例)	component BluetoothServer1 蓝牙服务端的一个实例

5. Notifier(通知)

Notifier 用来显示浮动信息、选择对话框、消息对话框和文本输入对话框。

附表 B. 42 Notifier 控件说明

模块名称	模块及模块说明	
AfterChoosing(事件)	when Notifier1.AfterChoosing choice do	在选择对话框中做出选择后触发本事件
AfterTextInput(事件)	when Notifier1.AfterTextInput response do	在文本输入对话框输入文本,并选择确定后触发本事件
LogError(方法)	call Notifier1.LogError message	显示错误信息
LogInfo(方法)	call Notifier1.LogInfo message	显示日志信息
LogWarning(方法)	call Notifier1.LogWarning message	显示警告信息
ShowAlert(方法)	call Notifier1.ShowAlert notice	显示临时通知,数秒钟后自动消失
ShowChooseDialog(方法)	call Notifier1.ShowChooseDialog message title button1Text button2Text cancelable	显示选择对话框
ShowMessageDialog(方法)	call Notifier1.ShowMessageDialog message title buttonText	显示消息对话框
ShowTextDialog(方法)	call Notifier1.ShowTextDialog message title cancelable	显示文本输入对话框
Notifier1(实例)	component Notifier1 通知的一个实例	

6. SpeechRecognizer(语音识别器)

SpeechRecognizer是一个非可视化控件,可以将语音输出转化为文本输出。

附表 B.43 SpeechRecognizer 控件说明

模块名称	模块及模块说明	
AfterGettingText(事件)	when SpeechRecognizer1.AfterGettingText result do	语音识别后事件
BeforeGettingText(事件)	when SpeechRecognizer1.BeforeGettingText do	语音识别前事件
GetText(方法)	call SpeechRecognizer1.GetText	将用户的语音数据转化为文字
Result(属性)	SpeechRecognizer1.Result 语音识别后的文字信息	
SpeechRecognizer1(实例)	component SpeechRecognizer1 语音识别器的一个实例	

7. TextToSpeech(语音转换器)

TextToSpeech是一个非可视化控件,可以将输入的文字转化为语音输出,可以通过设置语言种类及地区来设置输出语音。

附表 B.44 TextToSpeech 控件说明

模块名称	模块及模块说明	
AfterSpeaking(事件)	when TextToSpeech1.AfterSpeaking result do	语音转换后事件
BeforeSpeaking(事件)	when TextToSpeech1.BeforeSpeaking do	语音转换前事件
Speak(方法)	call TextToSpeech1.Speak message	将文字转化为语音输出

续表

模块名称	模块及模块说明
Country(属性)	set TextToSpeech1.Country to TextToSpeech1.Country 国家(地区)代码
Language(属性)	set TextToSpeech1.Language to TextToSpeech1.Language 语言代码
Result(属性)	TextToSpeech1.Result 输出的语音
TextToSpeech1(实例)	component TextToSpeech1 语音转换器的一个实例

8. TinyWebDB(微型网络数据库)

TinyWebDB是一个非可视化控件,可以通过网络完成对网络端数据库的存储和读取操作。

附表 B.45　TinyWebDB 控件说明

模块名称	模块及模块说明	
GotValue(事件)	when TinyWebDB1.GotValue tagFromWebDB valueFromWebDB do	当完成 GetValue 时触发本事件
ValueStored(事件)	when TinyWebDB1.ValueStored do	数据存储完成事件
WebServiceError(事件)	when TinyWebDB1.WebServiceError message do	网络数据库服务出错事件
GetValue(方法)	call TinyWebDB1.GetValue tag	从网络数据库服务器中获取给定标签为 tag 的数据

续表

模 块 名 称	模块及模块说明
StoreValue(方法)	call TinyWebDB1.StoreValue tag valueToStore 向网络数据库服务器中存储标签为 tag 的数据
ServiceURL(属性)	set TinyWebDB1.ServiceURL to TinyWebDB1.ServiceURL 微型网络数据库的 URL 值
TinyWebDB1(实例)	component TinyWebDB1 微型网络数据库模块的一个实例

9. Web(浏览器)

Web 控件是一个非可视化控件,可以完成对网页的访问及相关操作。

附表 B.46 Web 控件说明

模 块 名 称	模块及模块说明
GotFile(事件)	when Web1.GotFile url responseCode responseType fileName do SaveResponse 属性为 true 时,Get 方法的响应将产生文件,并引发此事件
GotText(事件)	when Web1.GotText url responseCode responseType responseContent do SaveResponse 属性为 false 时,Get 方法的响应将引发此事件
BuildPostDate(方法)	call Web1.BuildPostData list 将有两个子列的列表转换为关键字和值的数据
ClearCookies(方法)	call Web1.ClearCookies 清空 Cookies

续表

模块名称	模块及模块说明	
Get(方法)	call Web1.Get	执行一个 HTTP GET 请求,并根据属性 SaveResponse 获取响应。如果 SaveResponse 为 true,将响应保存成文件,并引发 GetFile 事件;如果 SaveResponse 为 false,将引发 Text 事件
HtmlTextDecode(方法)	call Web1.HtmlTextDecode htmlText	对 HTML 文本值进行解码
JsonTextDecode(方法)	call Web1.JsonTextDecode jsonText	对 Json 文本值进行解码
PostFile(方法)	call Web1.PostFile path	执行一个 HTTP POST 请求,如果 SaveRespose 属性值 true,响应将被保存在一个文件中并触发 GotFile 事件;如果 SaveResponse 属性值为 false 将触发 GotText 的事件
PostText(方法)	call Web1.PostText text	执行一个 HTTP POST 请求,使用 UFT-8 对字符进行编码。如果 SaveResponse 属性值为 true,将响应值保存在一个文件中并触发 GotFile 事件。如果 SaveResponse 属性值为 false,则触发 GotText 事件
PostTextWith Encoding(方法)	call Web1.PostTextWithEncoding text encoding	执行一个 HTTP POST 请求,用指定的方式对字符进行编码,如果 SaveResponse 属性值为 true,将响应值保存在一个文件中并触发 GotFile 事件。如果 SaveResponse 属性值为 false,则触发 GotText 事件
UriEncode(方法)	call Web1.UriEncode text	将字符转化为 Uri 编码方式
AllowCookies(属性)	set Web1.AllowCookies to / Web1.AllowCookies 是否将当前网页的相关信息存储到 Cookies 文件夹中	
RequestHeaders(属性)	set Web1.RequestHeaders to / Web1.RequestHeaders 返回的头信息	

续表

模 块 名 称	模块及模块说明
ResponseFileName（属性）	set Web1.ResponseFileName to Web1.ResponseFileName 返回信息的存储文件的位置信息
SaveResponse（属性）	set Web1.SaveResponse to Web1.SaveResponse 是否将返回的信息存储在一个文件中
Url（属性）	set Web1.Url to Web1.Url 访问网页的 Url 值
Web1（实例）	component Web1 浏览器的一个实例

B.9 Not ready for prime time(不成熟)控件

不成熟控件共 5 个，包括 FusiontablesContral、GameClient、SoundRecord、Voting 和 WebViewer，各个控件的事件、属性和方法如附表 B. 47～附表 B. 51 所示。

1. FusiontablesContral（融合表控制器）

融合表是谷歌文档的一部分，融合表控制器（FusiontablesContral）的主要功能是利用谷歌的 API 查询、创建和修改融合表。

附表 B. 47 FusiontablesContral 控件说明

模 块 名 称	模块及模块说明	
GotResult（事件）	when FusiontablesControl1.GotResult result do	当返回查询结果时触发的事件
DoQuery（方法）	call FusiontablesControl1.DoQuery	这个方法已经被弃用，被 SentQuery 取代
ForgetLogin（方法）	call FusiontablesControl1.ForgetLogin	当使用已经丢弃的用户名登录融合表时，强制进行重新认证

续表

模块名称	模块及模块说明
SendQuery(方法)	call FusiontablesControl1.SendQuery 执行查询融合表的操作
ApiKey(属性)	set FusiontablesControl1.ApiKey to FusiontablesControl1.ApiKey 谷歌 API 的 KEY
Query(属性)	set FusiontablesControl1.Query to FusiontablesControl1.Query 发送给谷歌融合表 API 的请求
FusiontablesControl1(实例)	component FusiontablesControl1 融合表控制器的一个实例

2. GameClient(游戏客户端)

GameClient 的主要功能是游戏服务器端的通信实现多人游戏。

附表 B.48 GameClient 控件说明

模块名称	模块及模块说明	
FunctionCompleted(事件)	when GameClient1.FunctionCompleted functionName do	当一个函数完成时所触发的事件
GotMessage(事件)	when GameClient1.GotMessage type sender contents do	当收到一个新的消息时触发的事件
Info(事件)	when GameClient1.Info message do	某一指定的消息发生时所触发的事件
InstanceIdChanged(事件)	when GameClient1.InstanceIdChanged instanceId do	当一个实例的 Id 属性发生变化时触发的事件

续表

模 块 名 称	模块及模块说明	
Invited(事件)	when GameClient1.Invited instanceId do	用户被邀请参加某一个游戏时触发的事件
NewInstanceMade(事件)	when GameClient1.NewInstanceMade instanceId do	创建一个新的实例时触发的事件
NewLeader(事件)	when GameClient1.NewLeader playerId do	游戏中被指定一个新的领导者时触发的事件
PlayerJoined(事件)	when GameClient1.PlayerJoined playerId do	有一个新的玩家加入游戏时触发的事件
PlayerLeft(事件)	when GameClient1.PlayerLeft playerId do	有一个玩家退出游戏时触发的事件
ServerCommand-Failure(事件)	when GameClient1.ServerCommandFailure command arguments do	当一个服务器命令失败时触发的事件
ServerCommand-Success(事件)	when GameClient1.ServerCommandSuccess command response do	当一个服务器命令成功地返回时触发的事件
UserEmailAddress-Set(事件)	when GameClient1.UserEmailAddressSet emailAddress do	当用户的电子邮件地址已经被设置时触发的事件
WebServiceError(事件)	when GameClient1.WebServiceError functionName message do	与网络服务通信发生错误时触发的事件

续表

模块名称	模块及模块说明
GetInstanceLists(方法)	call GameClient1.GetInstanceLists　返回游戏实例的列表
GetMessages(方法)	call GameClient1.GetMessages type count　获取指定的类型的消息
Invite(方法)	call GameClient1.Invite playerEmail　邀请指定的玩家到当前游戏
LeaveInstance(方法)	call GameClient1.LeaveInstance　玩家离开当前的游戏
MakeNewInstance(方法)	call GameClient1.MakeNewInstance instanceId makePublic　向服务器申请新建一个游戏的实例
SendMessage(方法)	call GameClient1.SendMessage type recipients contents　向收件人列表中的所有人发送消息
ServerCommand(方法)	call GameClient1.ServerCommand command arguments　向游戏服务器发送指定的命令
SetInstance(方法)	call GameClient1.SetInstance instanceId　设置实例的 ID 并加入指定的实例
SetLeader(方法)	call GameClient1.SetLeader playerEmail　设置指定的玩家为新的领导者
GameId(属性)	GameClient1.GameId 游戏的 ID
InstanceId(属性)	GameClient1.InstanceId 游戏的实例 ID,由 GameId 和 InstanceId 可以唯一确定一个游戏

续表

模 块 名 称	模块及模块说明
InvitedInstances（属性）	GameClient1.InvitedInstances 已经向当前的游戏玩家发送邀请的游戏实例的集合
JoinedInstances（属性）	GameClient1.JoinedInstances 当前的玩家已经加入的游戏实例的集合
Leader（属性）	GameClient1.Leader 游戏的领导者
Players（属性）	GameClient1.Players 当前游戏实例的在线玩家的集合
PublicInstances（属性）	GameClient1.PublicInstances 具有公开属性的游戏实例的集合
ServiceUrl（属性）	GameClient1.ServiceUrl 游戏服务的 URL
UserEmailAddress（属性）	set GameClient1.UserEmailAddress to GameClient1.UserEmailAddress 游戏玩家的电子邮件地址
GameClient1（实例）	component GameClient1 游戏客户端的一个实例

3. SoundRecorder（录音器）

SoundRecorder 的主要功能是音频录制。

附表 B.49 SoundRecorder 控件说明

模块名称	模块及模块说明	
AfterSoundRecorded（事件）	when SoundRecorder1.AfterSoundRecorded sound do	当音频文件录制完成后触发的事件
StartedRecording（事件）	when SoundRecorder1.StartedRecording do	开始录音时触发的事件
StoppedRecording（事件）	when SoundRecorder1.StoppedRecording do	停止录音时触发的事件
Start（方法）	call SoundRecorder1.Start	开始录制
Stop（方法）	call SoundRecorder1.Stop	停止录制
SoundRecorder1（实例）	component SoundRecorder1 录音器的一个实例	

4. Voting（投票）

Voting 的主要功能是进行网络投票，通过网络服务检索投票返回用户的选择。

附表 B.50 Voting 控件说明

模块名称	模块及模块说明	
GotBallot（事件）	when Voting1.GotBallot do	检索到一项投票时触发的事件
GotBallotConfirmation（事件）	when Voting1.GotBallotConfirmation do	收到服务器对投票的确认时触发的事件
NoOpenPoll（事件）	when Voting1.NoOpenPoll do	没有有效的投票主题时触发的事件

续表

模块名称	模块及模块说明
WebServiceError(事件)	when Voting1.WebServiceError message do 与网络服务通信时遇到错误时触发的事件
RequestBallot(方法)	call Voting1.RequestBallot 向指定的网络服务请求进行投票
SendBallot(方法)	call Voting1.SendBallot 向网络服务器发送一次完整的投票
BallotOptions(属性)	Voting1.BallotOptions 投票组件的选项列表
BallotQuestion(属性)	Voting1.BallotQuestion 投票组件的投票主题
UserEmailAddress(属性)	Voting1.UserEmailAddress 设备绑定的电子邮件地址(该属性已经被丢弃)
ServiceURL(属性)	set Voting1.ServiceURL to / Voting1.ServiceURL 投票服务的 URL
UserChoice(属性)	set Voting1.UserChoice to / Voting1.UserChoice 用户的投票选择
UserId(属性)	set Voting1.UserId to / Voting1.UserId 用于识别选民的 ID
Voting1(实例)	component Voting1 投票的一个实例

5. WebViewer(网页浏览器)

WebViewer 的主要功能是浏览网页,实现上网的功能。

附表 B.51　WebViewer 控件说明

模块名称	模块及模块说明
CanGoBack(方法)	call WebViewer1.CanGoBack　是否可以返回到浏览历史的上一页
CanGoForward(方法)	call WebViewer1.CanGoForward　是否可以进入到浏览历史的下一页
GoBack(方法)	call WebViewer1.GoBack　返回到浏览历史列表中的前一页
GoForward(方法)	call WebViewer1.GoForward　前进到浏览历史列表中的下一页
GoHome(方法)	call WebViewer1.GoHome　加载默认的主页
GoToUrl(方法)	call WebViewer1.GoToUrl url　加载指定的 URL
CurrentPageTitle(属性)	WebViewer1.CurrentPageTitle 正在浏览的网页的标题
CurrentUrl(属性)	WebViewer1.CurrentUrl 正在浏览的网页的 URL
FollowLinks(属性)	set WebViewer1.FollowLinks to　WebViewer1.FollowLinks 是否跟踪链接,如果跟踪则可以使用 GoForward 和 GoBack 功能
Height(属性)	set WebViewer1.Height to　WebViewer1.Height 组件高度
HomeUrl(属性)	set WebViewer1.HomeUrl to　WebViewer1.HomeUrl 组件的默认主页
Visible(属性)	set WebViewer1.Visible to　WebViewer1.Visible 设组件是否可见
Width(属性)	set WebViewer1.Width to　WebViewer1.Width 组件的宽度
WebViewer1(实例)	component WebViewer1 网页浏览器的一个实例